Technische Hochschule
Brandenburg
University of
Applied Sciences

Optimierung der Rechnungsdatenextraktion durch Einsatz von Large Language Models: Ansätze und Evaluation bei der aifinyo AG

Masterarbeit

zur Erlangung des Grades Master of Informatik

des Fachbereichs Informatik und Medien der

Technischen Hochschule Brandenburg

vorgelegt von:

Florian Pruß

Betreuer: Prof. Dr. Emanuel Kitzelmann

Zweitgutachter: Prof. Dr. Roland Fassauer

Brandenburg an der Havel, 18. September 2025

Kurzfassung

Die vorliegende Arbeit untersucht den Einsatz von Large Language Models (LLMs) zur Extraktion strukturierter Rechnungsdaten aus maschinenlesbaren PDF-Dokumenten im Kontext der aifinyo AG. Ziel ist es, die Genauigkeit und Robustheit moderner LLM-Ansätze mit der bestehenden OCR-basierten Lösung von Gini zu vergleichen, die derzeit im produktiven Einsatz ist. Dazu wird eine empirische Studie durchgeführt, in der verschiedene LLMs (Gemma, GPT-4, Claude) unter identischen Bedingungen getestet und mit manuell bereinigten Referenzdaten sowie historischen OCR-Ergebnissen verglichen werden. Der Fokus liegt auf der präzisen Extraktion geschäftskritischer Kernfelder wie Rechnungsnummer, Rechnungsdatum und Betrag. Ergänzend werden Strategien wie Zero-Shot-, Few-Shot- und Chain-of-Thought-Prompting sowie der Einfluss unterschiedlicher Textextraktionsbibliotheken untersucht, um Optimierungspotenziale und technische Machbarkeit zu bewerten. Die Ergebnisse zeigen, dass LLMs bei variablen Rechnungs-Layouts erkennbar bessere Erkennungsraten erzielen können und einen vielversprechenden Ansatz für die zukünftige Automatisierung der Rechnungsverarbeitung darstellen.

Schlüsselwörter

- Rechnungsdatenextraktion
- Large Language Models (LLM)
- Dokumentenverarbeitung
- Zero-Shot und Few-Shot Prompting
- Finanzautomatisierung / FinTech

Abstract

This thesis investigates the use of Large Language Models (LLMs) for extracting structured invoice data from machine-readable PDF documents in the context of aifinyo AG. The primary goal is to compare the accuracy and robustness of modern LLM-based approaches with the currently deployed OCR-based solution from Gini. An empirical study is conducted in which multiple LLMs (Gemma, GPT-4, Claude) are evaluated under identical conditions and benchmarked against manually curated reference data as well as historical OCR results. The focus is on accurately extracting business-critical key fields such as invoice number, invoice date, and total amount. Additionally, strategies like Zero-Shot, Few-Shot, and Chain-of-Thought prompting, as well as the impact of different text extraction libraries, are examined to assess optimization potential and technical feasibility. The findings demonstrate that LLMs achieve noticeably better extraction accuracy for invoices with variable layouts, indicating strong potential for future automation of invoice processing.

Keywords

- Invoice data extraction
- Large Language Models (LLMs)
- Document processing
- Zero-Shot and Few-Shot Prompting
- Financial process automation / FinTech

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AG	Aktiengesellschaft
Acc	Accuracy
API	Application Programming Interface
B2B	Business-to-Business
B2C	Business-to-Consumer
BERT	Bidirectional Encoder Representations from Transformers
CI	Confidence Interval / Konfidenzintervall
CID	Character Identifier (PDF-interne Zeichen-Referenz)
CMap	Character Map (Zuordnungstabelle zwischen PDF-internen Zeichen-IDs und Unicode-Zeichen)
CoT	Chain of Thought
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CSV	Comma-Separated Values
DI	Dependency Injection
ERP	Enterprise Resource Planning
FinTech	Financial Technology
GGUF	General Graph Universal Format (nur falls im Text genutzt)
GPU	Graphics Processing Unit
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KMU	Kleine und mittlere Unternehmen
KPI	Key Performance Indicator
LLM	Large Language Model
NLP	Natural Language Processing
ML	Maschinelles Lernen
OCR	Optical Character Recognition
OEM	OCR Engine Mode
OR	Odds Ratio
PDF	Portable Document Format
PSM	Page Segmentation Mode
p-Wert	Probability Value
RAG	Retrieval-Augmented Generation
RAM	Random Access Memory
SQL	Structured Query Language
UStG	Umsatzsteuergesetz
XML	Extensible Markup Language
Zero-/Few-Shot	Verfahren zur Modellanpassung ohne bzw. mit wenigen Beispielen
σ	Standardabweichung
χ^2	Chi-Quadrat

Abbildungsverzeichnis

Formellverzeichnis

Tabellenverzeichnis

1 Einleitung

Die fortschreitende Digitalisierung von Geschäftsprozessen führt in nahezu allen Branchen zu einer stetig wachsenden Menge an strukturierten und unstrukturierten Daten, die automatisiert verarbeitet werden können. Im Finanzdienstleistungsbereich nimmt die zuverlässige Extraktion von Rechnungsdaten eine zentrale Rolle ein, da sie die Grundlage für nachgelagerte Schritte wie Zahlungsfreigaben, Risikoprüfungen, Finanzbuchhaltung und Mahnwesen bildet. Fehlerhafte oder unvollständige Datenerkennung verursacht Verzögerungen, erhöht den manuellen Nachbearbeitungsaufwand und bringt finanzielle sowie rechtliche Risiken mit sich [1].

Moderne Large Language Models (LLMs) eröffnen neue Möglichkeiten der kontextsensitiven Verarbeitung natürlicher Sprache und bieten dabei eine höhere Flexibilität und Präzision als klassische Verfahren. Diese Flexibilität ist besonders für Unternehmen wie die aifinyo AG relevant, die mit den in Abschnitt 1.1 beschriebenen Herausforderungen konfrontiert sind.

Für diese Arbeit entstehen zwei Datensätze. Der erste ist ein gezielt kuratierter Developmentdatensatz mit besonders anspruchsvollen Rechnungsdokumenten, der zweite ein zufällig gezogener Evaluationsdatensatz, der das reale Belegaufkommen abbildet. Zentrales Werkzeug bildet eine eigens entwickelte Evaluationsplatform, die die automatisierte Durchführung und Auswertung von Extraktionsexperimenten ermöglicht. Mit ihr lassen sich verschiedene LLM-Strategien und Prompting-Varianten untersuchen und die Ergebnisse sowohl untereinander als auch im Vergleich zur bei aifinyo etablierten Rechnungs-OCR-Lösung von Gini analysieren. Die Evaluation verdeutlicht, dass LLMs die Genauigkeit bei der Extraktion der Kernfelder steigern und den Bedarf an manueller Nachbearbeitung verringern können, gleichzeitig jedoch neue Herausforderungen und Risiken entstehen.

Zunächst folgen die theoretischen Grundlagen und der Forschungskontext. Darauf baut die Beschreibung der im Rahmen dieser Arbeit entwickelten Evaluati-

onsplatform auf, über die die Experimente automatisiert durchgeführt und ausgewertet werden. Anschließend wird die konkrete Durchführung der Untersuchungen erläutert, bevor die zentralen Ergebnisse vorgestellt und abschließend eingeordnet sind.

1.1 Motivation und Problemstellung

Diese Arbeit entsteht in Zusammenarbeit mit der aifinyo AG, einem Berliner FinTech. Grundlage sind die dort verfügbaren Rechnungsinformationen sowie im Unternehmensalltag beobachtete Herausforderungen bei der Erfassung eingehender Rechnungen. Die aifinyo AG verarbeitet derzeit monatlich rund 15.000 Rechnungen auf Basis automatischer Texterkennung (OCR), ergänzt durch manuelle Validierung und Korrektur. Die Wirkung nachgelagerter, stark automatisierter Prozesse wird jedoch dadurch begrenzt, dass die Erfassung der Rechnungsdaten häufig unzuverlässig ist.

Hauptursachen sind uneinheitliche Layouts, komplexe Formatierungen und unterschiedliche Darstellungen von Beträgen (z. B. Brutto-/Nettosummen oder enthaltene Abschläge). Dies erfordert eine kontinuierliche manuelle Kontrolle und Korrektur und bindet ein festes Team von vier bis fünf Mitarbeitenden mit entsprechenden Kosten und verlängerten Durchlaufzeiten. Seit 2016 setzt das Unternehmen die Rechnungsdaten-Extraktionslösung der Gini GmbH ein, um zentrale Rechnungsfelder automatisiert zu erkennen.

Vor diesem Hintergrund untersucht die Arbeit, inwieweit Large Language Models (LLMs) eine präzisere und robustere Extraktion zentraler Rechnungsparameter ermöglichen und damit eine praktikable Alternative oder Ergänzung zu bestehenden OCR-basierten Lösungen darstellen.

1.2 Herausforderungen traditioneller OCR-basierter Systeme

Klassische OCR-Ansätze extrahieren Text primär über visuelle Mustererkennung und statische Layoutregeln, also sequenziell. Sie liefern bei standardisierten und klar strukturierten Belegen in der Regel zuverlässige Ergebnisse, stoßen jedoch

an Grenzen, wenn Layouts stark variieren, Feldbezeichnungen uneinheitlich sind oder komplexe Dokumentstrukturen vorliegen [2].

Auch maschinenlesbare PDFs mit eingebettetem Text sind nicht frei von Problemen, etwa wenn die interne Zeichenkodierung uneinheitlich ist oder die Segmentierung der Inhalte nicht der visuellen Struktur des Dokuments entspricht. Die Folgen sind Unstimmigkeiten in den extrahierten Daten, die sich in nachgelagerten Prozessen vervielfachen können. Besonders kritisch ist dabei, dass fehlerhafte, aber formal plausible Werte unbemerkt weiterverarbeitet werden können und so eine Dunkelziffer an verdeckten Prozessfehlern entsteht. Selbst kleine Erkennungsfehler können beispielsweise dazu führen, dass Rechnungen nicht automatisiert weiterverarbeitet werden oder in falsche Prozesspfade gelangen.

1.3 Potenzial von Large Language Models

LLMs bieten eine grundlegend andere Herangehensweise an die Extraktion von Rechnungsdaten. Sie verarbeiten Text nicht nur sequenziell, sondern kontextbasiert, und können Bedeutungszusammenhänge auch ohne explizite Layoutinformationen erkennen. Durch gezieltes Prompting lassen sich Modelle an unterschiedliche Dokumentvarianten anpassen, ohne für jede Layoutabweichung eigene Regeln oder Trainingsdaten bereitzustellen [3].

Erste Erfolge in verwandten Domänen wie dem Kundenservice [4], der maschinellen Übersetzung [5] oder der automatisierten Analyse juristischer Texte [6] zeigen, dass LLMs selbst komplexe semantische Abhängigkeiten zuverlässig erfassen können. Diese Anwendungsfelder sind zudem häufig stark regelbasiert und standardisiert, was die Übertragbarkeit auf strukturierte Extraktionsaufgaben unterstützt. Eine Anwendung für die standardisierte Erkennung zentraler Rechnungsparameter liegt daher nahe. So lassen sich unter anderem Rechnungsnummer, Rechnungsdatum, Bruttobetrag, Steuersätze und Debitor auch aus unterschiedlich strukturierten Dokumenten präzise extrahieren, ohne auf aufwendige Template- oder Regelpflege angewiesen zu sein.

2 Forschungsfrage und Zielsetzung

Die fortschreitende Entwicklung von LLMs eröffnet vollkommen neue Möglichkeiten für die automatisierte Dokumentenverarbeitung und stellt Unternehmen angesichts der steigenden Notwendigkeit digitaler Transformation vor die Frage, ob etablierte Rechnungs-OCRs auch künftig eine verlässliche Lösung darstellen oder ob LLM-basierte Ansätze in Zukunft erfolgversprechender sind. Während etablierte Dokumentenverarbeitungssysteme auf der Trennung von Textextraktion und semantischer Interpretation basieren, ermöglichen LLMs die kontextuelle Verarbeitung von Dokumenteninhalten ohne explizite Layout-Analyse.

Die vorliegende Arbeit beleuchtet dies anhand der Rechnungsverarbeitung bei der aifinyo AG. Im Mittelpunkt steht dabei die Frage, ob moderne LLM-Ansätze tatsächlich bessere Ergebnisse liefern als das seit Jahren verwendete Rechnungs-OCR von gini und welches Verbesserungspotenzial sich daraus nachweisen lässt.

Methodisch folgt die Untersuchung den etablierten Prinzipien empirischer Softwareengineering-Forschung. Dazu werden identische Rechnungsdokumente durch verschiedene LLM-basierte Ansätze verarbeitet und die daraus resultierenden Ergebnisse sowohl mit manuell geprüften Referenzdaten als auch mit den historischen Gini-OCR-Ergebnissen verglichen. Dieses Vergleichsdesign ermöglicht eine objektive Bewertung unter realistischen Geschäftsbedingungen und liefert sowohl quantitative Leistungsmessungen als auch Einblicke in praktische Implementierungsherausforderungen.

2.1 Primäre Forschungsfrage

Die beschriebene Ausgangslage führt zu einer zweigeteilten Forschungsfrage, die sowohl die grundsätzliche Auslesequalität von LLMs als auch deren praktischen Nutzen im direkten Vergleich untersucht:

F1: Wie präzise extrahieren Large Language Models strukturierte Daten aus maschinenlesbaren Rechnungs-PDFs?

F2: Inwiefern übertreffen sie dabei die durch das Rechnungs-OCR von gini erreichten Ergebnisse, und welche quantifizierbaren Performance-Unterschiede lassen sich empirisch nachweisen?

Diese zweigeteilte Fragestellung untersucht sowohl die Leistungsfähigkeit von LLMs als auch deren praktischen Nutzen im direkten Vergleich. Um methodisch saubere Vergleichsbedingungen zu schaffen, konzentriert sich die Evaluation auf textbasierte PDF-Dokumente. Diese Fokussierung eliminiert OCR-Textextraktionsfehler als Störvariable, da bei digital generierten PDFs der exakte Originaltext verfügbar ist, wodurch sich reine Interpretationsfehler von optischen Auslesefehlern trennen lassen.

2.1.1 Fokus auf Genauigkeitsverbesserung

Um diese Fragen beantworten zu können, liegt der Fokus vorwiegend auf der Extraktionsgenauigkeit als zentralem Bewertungskriterium. Konkret geht es darum, welcher Ansatz Rechnungsnummern, Rechnungsbetrag und Rechnungsdatum zuverlässiger erkennt und dabei weniger Fehler macht. Weitere Kriterien wie Kosten, Tokenverbrauch oder Verarbeitungszeit werden zwar erhoben, sind jedoch kein zentraler Bestandteil dieser Arbeit.

Die Evaluation folgt dabei dem Black-Box-Ansatz [7], wobei nur das Endergebnis zählt und die interne Funktionalität des Systems nicht berücksichtigt wird. Diese Herangehensweise gewährleistet somit faire Vergleichsbedingungen zwischen den verschiedenen LLM-basierten Ansätzen und den historischen Extraktionsdaten des Rechnungs-OCRs. Die Leistungsmessung erfolgt durch Ermittlung der Extraktionsgenauigkeit. Wie in Abbildung 2.1 gezeigt, stehen dabei die drei Kernfelder Rechnungsnummer, Rechnungsdatum und Rechnungsbetrag im Mittelpunkt. Dabei wird transparent dokumentiert, dass auch die Referenzdaten nicht fehlerfrei sind. Menschliche Prüfer machen ebenfalls Fehler, und manche Rechnungsinhalte sind tatsächlich mehrdeutig interpretierbar.

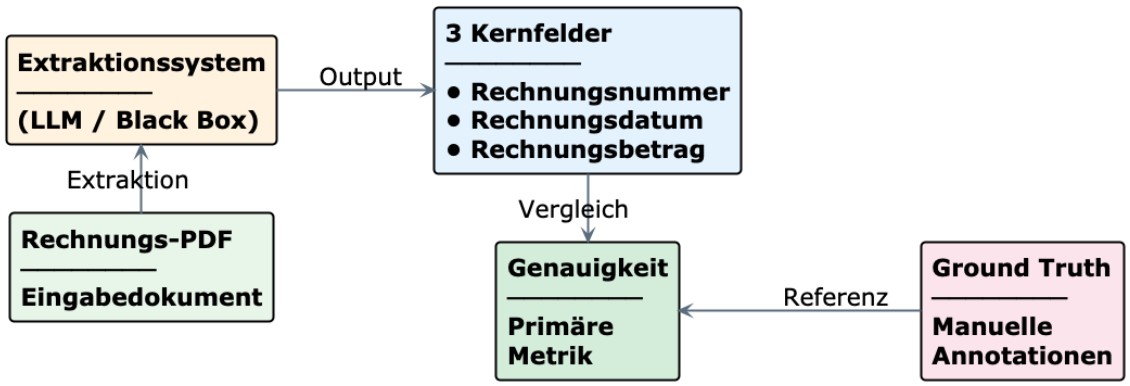


Abbildung 2.1 Black-Box-Ansatz zur Evaluation der Extraktionsgenauigkeit

2.1.2 Methodische Herausforderungen und Lösungsansätze

Eine zentrale Herausforderung dieser Arbeit liegt darin, dass die vorhandenen Infact-Referenzdaten menschliche Fehler, unterschiedliche Interpretationen und prozessbedingte Inkonsistenzen enthalten [8]. Für eine präzise Evaluation ist jedoch ein möglichst fehlerfreier Referenzdatensatz erforderlich.

Der Lösungsansatz verfolgt daher eine manuelle Bereinigung der Referenzdaten. Bei widersprüchlichen Fällen zwischen Infact-Daten und den besten verfügbaren Extraktionsergebnissen erfolgt eine manuelle Nachprüfung und Korrektur durch menschliche Experten. Dies gewährleistet die Erstellung eines qualitativ hochwertigen Basis-Referenzdatensatzes.

Weiterhin bestehen methodische Herausforderungen in Bezug auf die Konsistenz und Reproduzierbarkeit der LLM-Ausgaben. Selbst bei deterministischen Parametereinstellungen (z. B. $temperature = 0$) ist eine gewisse Ergebnisvarianz nicht auszuschließen [9]. Dies kann methodisch die Durchführung mehrerer Wiederholungsläufe pro Konfiguration erforderlich machen, um belastbare Aussagen zu ermöglichen. Zudem hängt die Qualität der Evaluationsdaten stark von der vorgelagerten PDF-Textextraktion ab, wobei sich fehlerhafte Segmentierung, inkonsistente Zeichencodierung oder unvollständige Erfassung unmittelbar auf die Leistung der Modelle auswirken [10]. Schließlich ergibt sich ein Trade-off zwischen Genauigkeit, Laufzeit und Kosten. Während leistungsstarke Modelle tendenziell bessere Ergebnisse erzielen, sind sie zugleich mit höherem Ressourcenverbrauch verbunden. Diese Faktoren werden zwar erhoben, bilden jedoch nicht den zentralen Schwerpunkt der Arbeit.

Diese Herangehensweise schafft eine solide Grundlage für objektive Vergleiche zwischen verschiedenen Extraktionsansätzen. Alle Bereinigungsschritte werden transparent dokumentiert, um die Nachvollziehbarkeit der Evaluationsergebnisse zu gewährleisten.

2.2 Spezifische Untersuchungsaspekte

Zur Beantwortung der Forschungsfragen (Abschnitt 2.1) wird die Untersuchung in vier komplementäre Analysebereiche gegliedert, die gemeinsam eine systematische Bewertung der Leistungsfähigkeit und Anwendbarkeit von LLMs für die Rechnungsdatenextraktion ermöglichen:

1. **Quantitative Performanceanalyse**

Im Fokus steht die Extraktionsgenauigkeit von LLMs im direkten Vergleich zu den historischen Gini-OCR-Ergebnissen. Bewertet wird die Korrektheit bei zentralen Rechnungsfeldern (Rechnungsnummer, -datum, Betrag) sowie die Gesamtperformance anhand der ermittelten Genauigkeit.

2. **Vergleichende Modellbewertung**

Verschiedene Modellfamilien (Gemma, Claude, GPT-4) werden unter identischen Bedingungen getestet. Ziel ist es, Unterschiede in Leistungsfähigkeit und Robustheit aufzuzeigen sowie Kriterien für die Modellauswahl abzuleiten.

3. **Untersuchung von Optimierungsstrategien**

Analysiert wird, inwiefern sich die Leistung von LLMs durch unterschiedliche Prompt-Engineering-Techniken steigern lässt. Dazu zählen Zero-Shot-, Few-Shot- und Chain-of-Thought-Prompting. Auf diese Weise werden systematisch Verbesserungspotenziale erfasst und praxisnahe Handlungsempfehlungen abgeleitet.

4. **Technische Implementierbarkeit**

Ergänzend werden nicht-funktionale Kriterien wie Tokenverbrauch, Verarbeitungszeiten und infrastrukturelle Anforderungen betrachtet. Diese Analyse erweitert die reine Performancebewertung um Aspekte der praktischen Umsetzbarkeit und bildet die Grundlage für eine Handlungsempfehlung im Kontext der aifinyo AG.

2.3 Wissenschaftliche Hypothesen

Basierend auf den definierten Untersuchungsbereichen werden die folgenden zentralen Hypothesenpaare für die empirische Evaluation aufgestellt. Die Nullhypothesen (H_0) werden im Verlauf der Studie überprüft, um sie gegebenenfalls zugunsten der entsprechenden Alternativhypothesen (H_A) zu verwerfen:

1. **Hypothesenpaar zur Layout-Robustheit**

   Nullhypothese (H_{01}): LLMs erzielen bei einer hohen Varianz unterschiedlicher Rechnungs-Layouts keine signifikant besseren Ergebnisse als das etablierte Rechnungs-OCR.

   Alternativhypothese (H_{A1}): LLMs erzielen bei einer hohen Varianz unterschiedlicher Rechnungs-Layouts signifikant bessere Ergebnisse als das etablierte Rechnungs-OCR.

2. **Hypothesenpaar zum Optimierungspotenzial**

   Nullhypothese (H_{02}): Der Einsatz von Prompt-Engineering (Zero-Shot, CoT und Few-Shot) führt zu keiner signifikanten Leistungssteigerung gegenüber einfachen LLM-Implementierungen.

   Alternativhypothese (H_{A2}): Der Einsatz von Prompt-Engineering (Zero-Shot, CoT und Few-Shot) führt zu einer signifikanten Leistungssteigerung gegenüber einfachen LLM-Implementierungen.

3. **Hypothesenpaar zur Generalisierbarkeit**

   Nullhypothese (H_{03}): LLMs zeigen bei unabhängigen Evaluationsdatensätzen eine signifikant schlechtere Leistung als bei den in der Optimierungsphase genutzten Entwicklungsdaten.

   Alternativhypothese (H_{A3}): LLMs zeigen bei unabhängigen Evaluationsdatensätzen eine vergleichbare oder bessere Leistung als bei den in der Optimierungsphase genutzten Entwicklungsdaten.

Die Ergebnisse der Untersuchung dienen dazu, diese Hypothesen zu überprüfen. Damit bilden sie sowohl die Grundlage für die wissenschaftliche Erkenntnisgewinnung als auch zentrale Entscheidungskriterien für eine spätere Einführung bei der aifinyo AG.

3 Anwendungskontext und Problemdomain

Die vorliegende Untersuchung findet im Umfeld der aifinyo AG [11] statt, einem inhabergeführten Unternehmen im Bereich der Finanztechnologie (FinTech) mit Sitz in Deutschland. Mit der Erfahrung aus über 13 Jahren unterstützt aifinyo kleine und mittelständische Unternehmen (KMU) bei ihrer Liquiditätsplanung. Um dies möglichst effizient und wirtschaftlich zu gestalten, werden vermehrt digitale und weitgehend automatisierte Lösungen für Rechnungsabwicklung, Finanzierung und Cashflow-Management eingesetzt. Dabei beschäftigt aifinyo rund 70 Mitarbeitende sowie eine eigene Softwareentwicklung, die maßgeblich zur kontinuierlichen Weiterentwicklung und Automatisierung der Geschäftsprozesse beiträgt.

Im Zentrum des Geschäftsmodells steht die automatisierte Verarbeitung eingehender Rechnungen im Rahmen der Rechnungsvorfinanzierung. Monatlich werden mehrere tausend Rechnungen verarbeitet. Dieses Volumen kann nur durch standardisierte und digitalisierte Abläufe effizient bewältigt werden. Der gesamte Prozess, beginnend mit der Erfassung und Validierung eingereichter Rechnungen über die Limitprüfung und Auszahlung bis hin zur Einzahlungszuordnung oder dem Forderungsmanagement, basiert auf der zuverlässigen Extraktion der Rechnungsinformationen. Fehlerhafte Daten, beispielsweise durch ungenaue Texterkennung, können zu Verzögerungen, falschen Zahlungen oder sogar zu Verstößen gegen regulatorische Vorgaben führen.

Die aifinyo AG misst der Weiterentwicklung der Rechnungsdatenextraktion daher zentrale Bedeutung zu, da sie wesentlich zur Effizienzsteigerung und Risikominimierung beiträgt. Dieses Kapitel beschreibt die organisatorischen und technischen Rahmenbedingungen, unter denen die Evaluation von LLM-basierten Methoden erfolgt. Es beleuchtet den softwaregestützten Rechnungsprozess der aifinyo AG, die rechtlichen Anforderungen an Rechnungen gemäß § 14 UStG sowie die Grenzen des derzeit eingesetzten Rechnungs-OCR, die den Ausgangspunkt für die Entwicklung und Bewertung neuer LLM-gestützter Ansätze bilden.

3.1 Rechnungsvorfinanzierung bei aifinyo

Der zentrale Geschäftsprozess der aifinyo AG ist die Verarbeitung eingereichter Rechnungen. Dabei stellt die Rechnungsvorfinanzierung [12] einen wesentlichen Teil des Hauptgeschäftes dar. In diesem Prozess werden offene Forderungen von den Forderungsinhabern (Kreditoren) angekauft, vorfinanziert und durch Abtretung des Zahlungsanspruches an aifinyo übertragen. Die angekauften Rechnungen werden dann nach Erreichen des Zahlungsziels vom ursprünglichen Rechnungsempfänger (Debitor) gegenüber aifinyo beglichen. Erfolgt keine Zahlung, wird die Forderung durch aifinyo als neuen Forderungsinhaber im Rahmen des Forderungsmanagements geltend gemacht.

Im Rahmen dieser Arbeit werden ausschließlich der Factoringprozess und die damit einhergehenden Herausforderungen betrachtet. Weitere Dienstleistungen der aifinyo AG wie die Einkaufsfinanzierung (Finetrading), Leasing oder Smart Billment, also das Schreiben und Verwalten eigener Rechnungen, können zwar von den Ergebnissen der Untersuchung profitieren, bleiben aber zunächst unberücksichtigt.

Der Fokus auf den Factoringprozess ergibt sich aus dem damit verbundenen, unternehmenseigenen ERP-System Infact, das diesen Prozess weitestgehend digitalisiert abbildet und somit eine umfassende und qualitativ hochwertige Datengrundlage für die Untersuchung schafft. Die Menge, Vollständigkeit und Genauigkeit der historischen Daten sind bei den weiteren Produkten nicht in diesem Umfang gegeben.

3.2 Die Rechnung als Ausgangspunkt

Auch wenn im wirtschaftlichen Sinne mit der Vorfinanzierung eine Forderung übernommen wird, beginnt der Prozess mit der zugrunde liegenden Rechnung. Sie bildet die rechtliche Grundlage für eine Forderung und enthält alle relevanten Informationen, die für die weitere Verarbeitung benötigt werden. Dabei unterliegt sie in Deutschland den Vorschriften des § 14 Umsatzsteuergesetzes [13], welches die Pflichtangaben sowie die Anforderungen an Echtheit, Unversehrtheit und Lesbarkeit von Rechnungen regelt. Für automatisierte Extraktionssysteme sind diese Vorgaben somit zentral, da sie die zu erreichende Mindestanforderung an die zu extrahierenden Felder darstellen. Zu den gesetzlich vorgeschriebenen Angaben zählen unter anderem Name und Anschrift des leistungserbringenden Unternehmers (Kreditor) und des Leistungsempfängers (Debitor), das Rechnungsdatum, eine fortlaufende Rechnungsnummer, die Art und Menge der Leistung, das Entgelt, der Steuersatz und der ausgewiesene Steuerbetrag.

Eine besondere Herausforderung stellt dabei die Identifikation der Rechnungsnummer dar. Gesetzlich vorgeschrieben sind lediglich ihre Eindeutigkeit und eine lückenlose Fortführung. Genaue Vorgaben zur Formatierung oder zum Aufbau der Rechnungsnummer existieren nicht. In der Praxis sind daher vielfältige Versionen zu finden. Etwa rein numerische Formate (z. B. „123456789"), alphanumerische Formate (z. B. „R-001 / 2025") oder Kombinationen mit Datumsangaben (z. B. „20250627-001") sind möglich. Eine Rechnungsnummer wie „R-001 / 2025" kann somit ebenso in der Form „R-001/2025" korrekt sein, was speziell in Hinblick auf optische Zeichenerkennung (OCR) eine Herausforderung darstellt.

Für FinTech-Unternehmen wie die aifinyo AG ist eine präzise und nachvollziehbare Erfassung dieser Informationen von hoher Bedeutung. Es handelt sich um Rechnungsdaten, die sowohl für die anschließende interne Verwaltung der Forderungen als auch für die Honorierung und Annahme sowie für die Zahlungszuordnung entscheidend sind. Bereits kleine Fehler können hierbei weitreichende Folgen nach sich ziehen.

3.3 Ablauf der Rechnungsverarbeitung

Die Verarbeitung von Rechnungen erfolgt bei aifinyo in einem weitgehend digitalisierten Prozess, beginnend mit der Erstellung oder dem Upload der Rechnung durch die Kunden im aifinyo-Kundenportal. Die Rechnungen werden als PDF oder Bilddatei (PNG, JPEG) in die unternehmenseigene ERP-Software Infact hochgeladen. Anschließend erfolgt eine automatisierte Verarbeitung durch das Rechnungs-OCR der gini GmbH, bevor geschulte Kundenbetreuer die Ergebnisse kontrollieren und gegebenenfalls korrigieren. Die so validierten Rechnungsdaten bilden die Grundlage für die nachfolgenden, teilweise automatisierten Prozessschritte.

Diese umfassen unter anderem die Limitprüfung, die Veritätsprüfung, die Auszahlung, die Zahlungszuordnung sowie die Buchung und den Abschluss. Im Anschluss daran folgt das Forderungsmanagement. Wie Abbildung 3.1 zeigt, sind diese Schritte eng miteinander verknüpft und können weitere Folgeprozesse wie automatische Buchungen, E-Mails oder zusätzliche Prüfungen auslösen. Eine präzise Rechnungsdatenextraktion ist daher von grundlegender Bedeutung, da Fehler weitreichende wirtschaftliche Folgen nach sich ziehen können – von fehlerhaften Auszahlungen bis hin zu Forderungsausfällen oder Reputationsschäden.

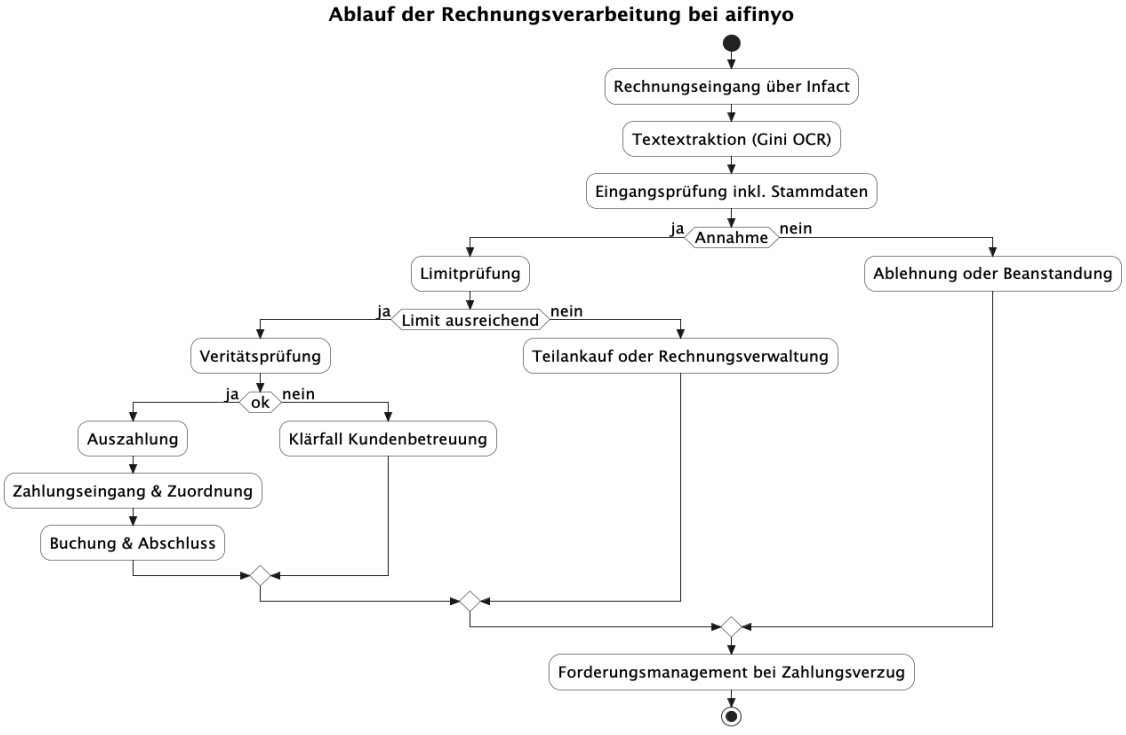


Abbildung 3.1 Ablauf der Rechnungsverarbeitung

Im Folgenden werden die einzelnen Prozessschritte der Rechnungsverarbeitung bei aifinyo detaillierter beschrieben, wobei insbesondere die Implikationen von Extraktionsfehlern aufgezeigt werden, ohne jedoch vertieft auf juristische oder wirtschaftliche Aspekte einzugehen.

Limitprüfung

Nach dem Auslesen der Rechnungsdaten und ihrer Validierung erfolgt zunächst eine Limitprüfung. Hierbei wird geprüft, ob der für die Vorfinanzierung der Rechnung nötige Ankaufrahmen der Kundenbeziehung ausreichend ist. Grundlegend hierfür sind die Risikobewertungen der Risikoabteilung und die daraus resultierenden Ankaufrahmen zwischen Kreditoren und Debitoren sowie individuell vertraglich zugesicherte Ankaufrahmen. Das Limitmodul übernimmt auf Basis der vorab definierten Ankaufrahmen automatisch die Ankaufentscheidung und erfordert nur ein manuelles Einschreiten, wenn vorab definierte Schwellwerte erreicht sind oder Auffälligkeiten wie beispielsweise stark abweichende Rechnungsbeträge auftreten. Fehlerhafte Rechnungsbeträge oder falsch ausgelesene Debitoren können hierbei dazu führen, dass es zu unnötigen Verzögerungen, zur Ablehnung des Rechnungsankaufs oder sogar zur Ausnutzung fremder Ankaufrahmen kommt.

Veritätsprüfung

Nach der Limitprüfung erfolgt die Veritätsprüfung. Dabei wird geprüft, ob die abgerechnete Leistung gegenüber dem Debitor erbracht wurde und somit die Werthaltigkeit der Forderung besteht. Je nach Risikoeinstufung der Rechnungsbeziehung erfolgen hierfür stichprobenartige Prüfungen in unterschiedlicher Intensität. Hierzu werden in erster Linie automatisierte Rückfragen per E-Mail an die Debitoren versendet, in denen die Bestätigung der Leistungserbringung erbeten wird. Falsch erfasste Rechnungsdaten wie Rechnungsnummer, Rechnungsbetrag oder Debitoren führen hier ebenfalls zu Nachfragen und somit zu manueller Nacharbeit beziehungsweise zur Preisgabe von Daten nicht involvierter dritter Parteien.

Auszahlung

Die Auszahlung erfolgt auf Grundlage einer buchhalterischen Verrechnung der Kreditorenkonten. Hierbei wird der Saldo des jeweiligen Kreditorenkontos geprüft, wobei sämtliche Buchungsvorgänge seit der letzten Verrechnung berücksichtigt werden. Ergibt sich daraus ein positiver Saldo zugunsten des Kreditors, wird automatisch eine Überweisungsbuchung an das hinterlegte Referenzkonto des Kunden gebucht und nach Freigabe eine Überweisung ausgelöst. Fehlerhafte Rechnungsbeträge oder nicht korrekt zugeordnete Debitoren haben hierbei falsche Buchungen zur Folge, die im Sinne einer ordentlichen Buchführung storniert und erneut korrekt gebucht werden müssen. Solche Korrekturen sind auf der Abrechnung des Kunden ersichtlich und können ebenfalls zu Nachfragen und manuellem Mehraufwand führen.

Zahlungszuordnung

In der logischen Abfolge des Rechnungsverarbeitungsprozesses schließt sich an die Auszahlung die Zahlungszuordnung an. Auch wenn sie als prozessual nachfolgender Schritt verstanden werden kann, handelt es sich in der Praxis um eine parallele Aufgabe, da eingehende Zahlungen der Debitoren täglich eingehen und zugeordnet werden müssen. Die Zuordnung erfolgt dabei weitestgehend teilautomatisiert. Anhand von Merkmalen wie bereits bekannten Bankverbindungen, den für den Zahlungseingang vorgesehenen Eingangskonten, der Rechnungsnummer im Verwendungszweck und dem Rechnungsbetrag wird der Kundenbetreuung ein Vorschlag zur Zahlungszuordnung gemacht. Liegen diese Informationen unvollständig, fehlerhaft oder inkonsistent vor, kann das System keine eindeutige Zuordnung vornehmen. In solchen Fällen ist eine manuelle Prüfung erforderlich, die mit erheblichem Aufwand verbunden sein kann.

Forderungsmanagement

Bleibt eine Zahlung nach Erreichen der Fälligkeit aus, beginnt der Prozess des Forderungsmanagements. Dieser ist im Wesentlichen automatisiert und umfasst ein mehrstufiges Verfahren, das mit der Zahlungserinnerung beginnt und über Mahnungen und Eskalationsschreiben mit dem Inkassoprozess endet. Die Grundlage für diesen Ablauf bilden die im System gespeicherten Rechnungsinformationen. Fehlerhaft erfasste Fälligkeiten (also Rechnungsdatum + Zahlungsziel), Rechnungsbeträge oder Rechnungsnummern sowie Debitorendaten können zu weitreichenden Problemen führen. Ein falsch erkanntes Zahlungsziel kann zu einer verfrühten Mahnung führen und die Kundenbeziehung unnötig belasten. Unstimmigkeiten bei Rechnungsnummer oder Betrag gefährden die rechtliche Grundlage der Forderung, während fehlerhafte Debitorenzuordnungen sogar zu Mahnschreiben an unbeteiligte Dritte führen können. Solche Vorfälle untergraben die Professionalität des Unternehmens und erfordern eine aufwändige manuelle Nachbearbeitung.

3.4 Infact als zentrale Softwareplattform

Die in Abschnitt 3.3 dargestellten Schritte der Rechnungsverarbeitung werden innerhalb der aifinyo AG vollständig durch die unternehmenseigene ERP-Software Infact abgebildet. Sie wurde speziell für die Anforderungen des Factorings entwickelt und bündelt alle relevanten Abläufe von der Einreichung der Rechnung bis hin zum Forderungsmanagement in einer zentralen Plattform.

Aus Sicht der Kunden reduziert sich die Nutzung im Wesentlichen auf zwei Funktionen. Die Einreichung neuer Rechnungen sowie das Dashboard, das den aktuellen Status der eingereichten Forderungen transparent macht. Die eigentliche Verarbeitung der Rechnungen erfolgt dagegen fast ausschließlich durch interne Rollen. Kundenbetreuung und Risikoabteilung nutzen Infact, um die aus der Texterkennung gewonnenen Rechnungsdaten zu prüfen, zu korrigieren und für die nachfolgenden Prozessschritte freizugeben. Damit liegt der Schwerpunkt der Software weniger auf der Kundenschnittstelle, sondern vielmehr auf der internen Abbildung, Steuerung und Dokumentation des gesamten Workflows.

Besondere Bedeutung haben dabei die Validierungsschritte und die Geschäftspartnerlogik. Rechnungen können nur durch registrierte Kreditoren eingereicht und ausschließlich gegen Debitoren adressiert werden, die im Adressbuch des Kreditors hinterlegt sind. Diese Adressdaten werden nach der ersten Einreichung durch Kundenbetreuung und Risiko verifiziert, sodass Felder wie Kreditor oder Debitor im weiteren Prozess weniger fehleranfällig sind. Relevanter für die hier untersuchte Datenextraktion sind dagegen Felder wie Rechnungsnummer, Datum und Betrag, deren Genauigkeit unmittelbaren Einfluss auf Limitprüfung, Auszahlung, Zahlungszuordnung und Forderungsmanagement hat.

Infact übernimmt somit eine Doppelfunktion. Einerseits stellt es für die Kunden die Schnittstelle zur Einreichung und Statusabfrage bereit, andererseits bildet es als interne Plattform die Grundlage für alle faktorrelevanten Arbeitsschritte. Für die vorliegende Untersuchung ist es von zentraler Bedeutung, da sämtliche Rechnungen, die in den Factoringprozess eingehen, in Infact gespeichert werden und damit die Datenbasis für die Evaluation bilden.

4 Stand der Forschung und technische Grundlagen

Die automatisierte Extraktion von Informationen aus Geschäftsdokumenten stellt ein kontinuierlich wachsendes Forschungsgebiet dar [14], [15], das durch den zunehmenden Einsatz von maschinellem Lernen und zuletzt von Large Language Models neue methodische Impulse erhält. Dieses Kapitel gibt einen Überblick über die wesentlichen Forschungsrichtungen und technischen Grundlagen, die für die Bewertung moderner LLM-basierter Extraktionsstrategien relevant sind.

Die Methoden zur automatisierten Informationsextraktion haben sich von regelbasierten Ansätzen über maschinelles Lernen bis hin zu Large Language Models kontinuierlich weiterentwickelt. Diese Entwicklung prägt das heutige Verständnis von Dokumentenverarbeitung und Rechnungsdatenextraktion.

Ein weiterer zentraler Aspekt in dieser Arbeit ist die Rolle des PDF-Formats als technische Grundlage der digitalen Dokumentenverarbeitung. Die Eigenschaften dieses Formats und die Vielfalt seiner Generierungsprozesse schaffen sowohl Möglichkeiten als auch Herausforderungen für automatisierte Extraktionsverfahren.

Die aktuellen Entwicklungen im Bereich der Large Language Models eröffnen neue Paradigmen für die Dokumentenverarbeitung, die über traditionelle OCR-basierte Ansätze hinausgehen. Diese theoretischen und methodischen Grundlagen bilden den Rahmen für das Verständnis moderner Extraktionsarchitekturen und deren Potenziale.

4.1 PDF als Medium der Rechnungsverarbeitung

Das Portable Document Format (PDF) wurde 1993 von Adobe im Rahmen des sogenannten Camelot-Projekts eingeführt, mit dem Ziel, ein plattformunabhängiges, layoutgetreues Austauschformat für elektronische Dokumente zu schaffen [16]. Die Spezifikation wurde 2008 erstmals als offener Standard unter ISO 32000-1 normiert und 2017 mit ISO 32000-2 aktualisiert. Seither gilt PDF als Defacto-Standard für den elektronischen Dokumentenaustausch in Verwaltung und Wirtschaft.

Die technischen Eigenschaften von PDF-Dokumenten sind ein relevanter Aspekt für das Verständnis automatisierter Informationsextraktionsverfahren. PDF wurde ursprünglich als finales Ausgabemedium konzipiert und gewährleistet eine konsistente visuelle Darstellung, bildet jedoch die logische Struktur der enthaltenen Informationen nur eingeschränkt ab [17]. Zwar wurden mit tagged PDFs ab Version 1.4 erste Ansätze zur semantischen Auszeichnung eingeführt, diese sind jedoch in der Praxis uneinheitlich implementiert und häufig unvollständig. Eine aktuelle Analyse zur PDF-Barrierefreiheit zeigt beispielsweise, dass weniger als 3,2 % aller untersuchten PDFs alle strukturellen Kriterien erfüllen, während fast 75 % keinerlei Tags enthalten. Zudem bestehen für viele PDF-Generatoren erhebliche technische Hürden bei der automatischen Erzeugung korrekt getaggter Dokumente [18].

Unterschiede in den PDF-Erstellungsmethoden, etwa zwischen nativ generierten, getaggten, nicht getaggten oder bildbasierten PDFs, können die Qualität der Textextraktion erheblich beeinflussen und damit nachgelagerte Verarbeitungsprozesse erschweren oder verändern.

4.1.1 Generierte PDFs mit eingebettetem Text

Die Mehrheit der bei der aifinyo AG verarbeiteten Rechnungsdokumente entstammt automatisierten PDF-Generierungsprozessen, bei denen Daten aus Enterprise-Resource-Planning-Systemen (ERP) oder Buchhaltungssystemen in strukturierte Dokumentenlayouts überführt werden. Diese maschinengenerierten PDFs enthalten den Rechnungstext bereits in digitaler, durchsuchbarer Form und erfordern theoretisch keine optische Zeichenerkennung (OCR) im klassischen Sinne.

Die Extraktion von Textinhalten aus solchen Dokumenten wird jedoch durch deren interne Struktur erschwert, da diese nicht der visuellen Anordnung entspricht [19]. Textfragmente können in beliebiger Reihenfolge gespeichert sein, Tabellenstrukturen liegen häufig als separate Textblöcke ohne explizite Relationsinformationen vor, und semantische Zusammenhänge zwischen Datenelementen müssen aus der räumlichen Anordnung der Koordinaten abgeleitet werden.

Ein besonderes Phänomen stellen dabei CID-Zeichen dar. Sie entstehen, wenn PDF-Generatoren proprietäre oder nicht standardkonforme Schriftarten verwenden und Zeichen intern über Character Identifiers (CIDs) adressieren. Ohne eine korrekt interpretierbare Zuordnungstabelle (CMap) können diese CIDs bei der Textextraktion nicht in Unicode-Zeichen übersetzt werden. Das führt dazu, dass statt lesbarem Text unverständliche Platzhalter wie (cid:61) extrahiert werden [20].

Zur Extraktion solcher Inhalte werden in der Praxis spezialisierte Bibliotheken wie Apache PDFBox [21], PDFMiner [22] oder PDF-Reader [23] eingesetzt. Diese greifen direkt auf die im PDF gespeicherten Objekte zu und rekonstruieren die darin eingebetteten Textfragmente, ohne dass externe Verfahren wie OCR notwendig sind. Dabei gehen jedoch wesentliche strukturelle Informationen verloren. Tabellen oder mehrspaltige Layouts werden zu einer linearen Abfolge von Textzeilen reduziert und die visuelle Hierarchie des Dokuments wandelt sich in eine flache Textdarstellung. Abbildung 4.1 verdeutlicht diesen Strukturverlust anhand des Vergleichs zwischen Original und extrahiertem Textlayer.

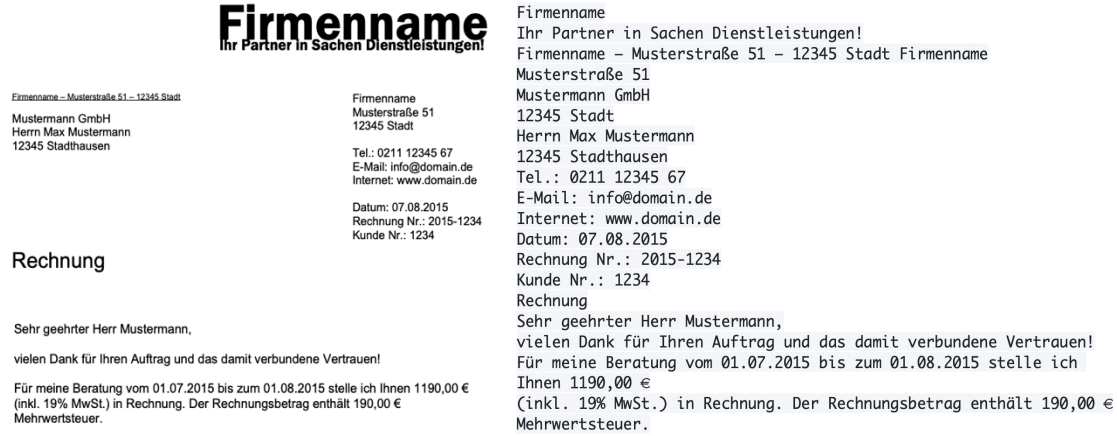


Abbildung 4.1 PDF-Textextraktion - Strukturverlust vom Original zum Textlayer

Diese strukturellen Brüche innerhalb funktionaler Layoutbereiche verdeutlichen die Grenzen klassischer Parsing-Verfahren. Damit zeigt sich, dass rein regelbasierte Ansätze die inhärenten Strukturprobleme nur unzureichend ausgleichen können. Hier eröffnen sich neue Perspektiven durch den Einsatz semantisch trainierter Modelle. Die Interpretation und korrekte Zuordnung extrahierter Fragmente kann durch kontextsensitives Post-Processing, etwa mit modernen NLP-Modellen, verbessert werden. Large Language Models (LLMs) können hier zusätzlich Potenzial zeigen, da sie auch bei fehlender struktureller Information in der Lage sind, semantische Zusammenhänge zu erkennen und implizite Bezüge zwischen Fragmenten herzustellen [24].

4.1.2 Bildbasierte PDFs

Eine weitere Kategorie von Rechnungsdokumenten entsteht durch die Digitalisierung physischer Belege mittels Scanner oder Smartphone-Kameras, wobei die resultierenden Bilddateien anschließend in PDF-Container eingebettet werden. Diese bildbasierten PDFs können sowohl als reines PDF ohne Textinformationen vorliegen als auch als Hybrid-Dokumente, die zusätzlich zur visuellen Repräsentation eine durch nachgelagerte OCR-Verarbeitung generierte Textebene enthalten. Diese Textebene wird dabei typischerweise durch OCR-Software wie beispielsweise Tesseract eingebettet, um dem Nutzer ein wie für normale Textdokumente gewöhnliches Kopieren zu ermöglichen [25].

Die Herausforderungen bei der Verarbeitung bildbasierter PDFs sind vielschichtig und beginnen bereits bei der Qualität der visuellen Erfassung. Ungleichmäßige Beleuchtung, Aufnahmewinkel, Schatten oder Reflexionen können die nachgelagerte OCR-Performance erheblich beeinträchtigen und zu Erkennungsfehlern führen [26]. Moderne Smartphone-Kameras und auf Dokumentenerfassung spezialisierte Scanner-Apps haben diese Problematik zwar durch automatische Bildkorrekturen und Kontrastanpassungen reduziert, dennoch bleibt die Qualitätsvarianz ein signifikanter Faktor für die Zuverlässigkeit automatisierter Extraktionsprozesse.

Die in bildbasierten PDFs eingebettete OCR-Textschicht neigt in der Praxis zu Fehlern, da sie auf der Interpretation visueller Zeichen basiert. Der Erkennungsprozess ist naturgemäß anfällig für Verwechslungen, etwa zwischen ähnlich aussehenden Zeichen wie „0" und „O" oder „1" und „l" [27]. Neben typografischen Ähnlichkeiten können auch Bildrauschen, ungleichmäßige Ausleuchtung oder niedrige Auflösung zu inkonsistenten oder fragmentierten Textergebnissen führen [28]. Solche zusätzlichen Unsicherheiten erschweren die zuverlässige maschinelle Weiterverarbeitung erheblich. Für die in dieser Arbeit untersuchte Rechnungsdatenextraktion bedeutet dies, dass bildbasierte PDFs zusätzliche Unsicherheiten einführen, die unabhängig von den Fähigkeiten der eingesetzten LLMs bestehen.

4.1.3 PDF-Parsing

Die automatisierte Extraktion strukturierter Informationen aus PDF-Dokumenten erfordert spezialisierte Tools, die unterschiedliche Ansätze zur Bewältigung der inhärenten Strukturherausforderungen verfolgen. Eine vergleichende Analyse verschiedener PDF-Parsing-Bibliotheken zeigt erhebliche Unterschiede in ihrer Fähigkeit, mit den diversen PDF-Generierungsmustern und Layoutkomplexitäten umzugehen [10].

Eine umfassende Vergleichsstudie von Adhikari und Agarwal (2024) zeigt, dass sich PDF-Parsing-Bibliotheken in ihrer Eignung für unterschiedliche Dokumentkategorien erheblich unterscheiden. In ihrer Analyse von zehn populären Tools anhand des DocLayNet-Datensatzes erzielten PyMuPDF und pypdfium die besten Ergebnisse bei standardisierten, tabellarischen Layouts wie bei Finanzdokumenten [19].

Spezialisierte Tabellen-Extraktions-Tools wie Camelot [29] oder Tabula [30] versuchen, die räumlichen Beziehungen zwischen Textfragmenten zu rekonstruieren und strukturierte Tabellendaten zu generieren. Diese Ansätze stoßen jedoch bei komplexen oder unregelmäßigen Layouts schnell an ihre Grenzen, da sie auf Heuristiken zur Erkennung von Zellen- und Spaltengrenzen angewiesen sind [31].

Trotz der aufgezeigten Limitationen stellen PDF-Parsing- und Tabellen-Extraktions-Tools eine essentielle Vorverarbeitungsstufe dar. Sie liefern die strukturierten Rohdaten, die als Eingabe für LLM-basierte Verfahren dienen und damit die Grundlage für deren weiterführende semantische Analyse und Interpretation bilden.

4.1.4 Strukturierte Rechnungsformate

Mit der Einführung der EU-Norm EN 16931 und Formaten wie X-Rechnung und ZUGFeRD existieren standardisierte Rechnungsformate, die Rechnungsinhalte in maschinenlesbarer Form (XML-basiert) bereitstellen und das Problem der Textextraktion theoretisch erheblich reduzieren könnten [32], [33]. In Deutschland ist der Empfang elektronischer Rechnungen im X-Rechnungsformat für bestimmte Unternehmen seit dem 01.01.2025 verpflichtend. Eine allgemeine Verpflichtung zur Ausstellung tritt jedoch erst ab 2027 in Kraft. Zudem bleiben große Teile des B2C-Marktes und zahlreiche internationale Transaktionen vorerst unberührt.

In der Praxis dominiert daher weiterhin das PDF als Rechnungsmedium, insbesondere im internationalen Kontext, sodass die automatisierte Extraktion aus unstrukturierten Dokumenten weiterhin ein relevantes Forschungsthema bleibt.

4.2 Klassische Ansätze der Rechnungsdatenextraktion

Die Forschungsgeschichte der automatisierten Rechnungsdatenextraktion ist durch einen evolutionären Entwicklungsprozess geprägt, der die Grundlage für heutige Verfahren geschaffen hat (Abbildung 4.2). Diese klassischen Ansätze haben sowohl methodische Erkenntnisse als auch praktische Limitationen hervorgebracht, die den Kontext für moderne Entwicklungen bilden.

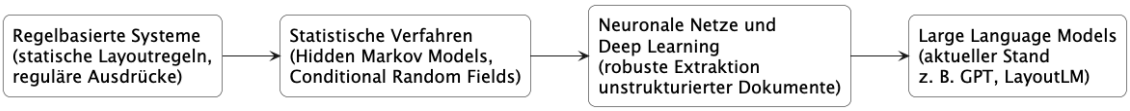


Abbildung 4.2 Entwicklung der Rechnungsdatenextraktionsverfahren

Die wissenschaftliche Entwicklung lässt sich in verschiedene Paradigmen unterteilen, die jeweils spezifische Beiträge zum Forschungsfeld geleistet haben. Frühe regelbasierte Systeme arbeiteten mit statischen Layoutregeln und regulären Ausdrücken, erzielten bei klar strukturierten Belegen zuverlässige Ergebnisse, scheiterten jedoch an variablen Layouts [34]. Spätere statistische Verfahren, etwa Hidden Markov Models oder Conditional Random Fields, verbesserten die Generalisierbarkeit und ermöglichten eine probabilistische Modellierung komplexerer Layouts [35]. Mit dem Aufkommen neuronaler Netze und Deep-Learning-Methoden wurde schließlich die Grundlage für die aktuelle Forschung gelegt, in der Rechnungsinformationen zunehmend robust aus unstrukturierten Dokumenten extrahiert werden können.

Diese historische Perspektive ist Grundlage für das Verständnis aktueller Forschungstrends und ermöglicht eine fundierte Einordnung moderner Ansätze in den breiteren wissenschaftlichen Kontext der Dokumentenverarbeitung.

4.2.1 OCR-basierte Verfahren und Post-Processing-Strategien

Optische Zeichenerkennung (OCR) bildet seit Jahrzehnten die Grundlage der automatisierten Textextraktion aus gescannten Dokumenten. Systeme wie Tesseract [25] haben erhebliche Fortschritte in der Erkennung von gedrucktem Text erzielt, etwa durch adaptive Klassifikatoren sowie Verfahren zur Korrektur von Schiefstellungen und gekrümmten Textzeilen. Dennoch bestehen Schwächen, insbesondere bei komplexen Layouts, Tabellenstrukturen und proportionalem Text. Diese Herausforderungen treten auch in rechnungsähnlichen Dokumenten auf, die durch variierende Layouts und Sprachmischungen eine hohe Fehleranfälligkeit für OCR-basierte Verfahren aufweisen [36]. Um die inhärenten Fehlerquellen der OCR zu kompensieren, werden vielfältige Post-Processing-Techniken eingesetzt.

Die automatisierte Rechnungsverarbeitung basiert traditionell auf einem zweistufigen Verfahren. Zunächst wird mittels OCR der visuelle Inhalt in maschinenlesbaren Text überführt, wonach anschließend eine regelbasierte Interpretation der extrahierten Inhalte erfolgt. Typische Post-Processing-Strategien [37] umfassen:

- **Pattern Matching**: Abgleich extrahierter Zeichenfolgen mit charakteristischen Schlüsselbegriffen wie „Rechnungsnummer", „Betrag" oder „Datum".
- **Layout-Heuristiken**: Analyse der relativen Position und räumlichen Anordnung von Textelementen.
- **Formatvalidierung**: Überprüfung extrahierter Werte gegen definierte Datums-, Betrags- oder Referenznummernmuster.

Diese Ansätze liefern bei standardisierten Rechnungsformaten akzeptable Erkennungsraten, sind jedoch sehr anfällig für abweichende Layouts oder unkonventionelle Formatierungen. Ihre Wartung erfordert kontinuierliche manuelle Anpassung und skaliert schlecht für heterogene Dokumentensammlungen.

Neuere Arbeiten erweitern diese klassische Pipeline um modulare Vorverarbeitungsschritte [1]. So werden Dokumente vor der OCR beispielsweise entzerrt, Artefakte wie Linien oder Barcodes entfernt und Layoutinformationen gezielt extrahiert, bevor die regelbasierte Analyse erfolgt. Dies steigert die Robustheit gegenüber variierenden Rechnungsformaten und verbessert die Erkennungsraten insbesondere bei realen, qualitativ schwankenden Dokumentstapeln.

Parallel dazu gewinnen „Machine-Learning"-basierte Ansätze zunehmend an Bedeutung. Anstelle vordefinierter Regeln werden trainierte Modelle eingesetzt, die relevante Felder direkt erkennen oder Layoutinformationen kontextsensitiv interpretieren. Ein Beispiel ist das templatefreie System CloudScan [38], das zeigt, dass solche Verfahren eine höhere Generalisierbarkeit gegenüber variierenden Rechnungsformaten ermöglichen. Allerdings erfordern sie umfangreiche annotierte Trainingsdaten und zeigen bei sehr unstrukturierten Dokumenten teils noch inkonsistente Ergebnisse.

4.2.2 Industrielle Lösungen

Der kommerzielle Markt für Rechnungsdatenextraktion wird von spezialisierten Anbietern dominiert, die proprietäre Kombinationen aus OCR, Machine Learning und regelbasierten Post-Processing-Mechanismen einsetzen. Anbieter wie Gini, Klippa, Mindee oder ABBYY richten ihre Systeme branchenüblich auf wiederkehrende Dokumentstrukturen aus, was eine gezielte Optimierung auf bestimmte Layout- und Formatvarianten ermöglicht.

Gini als in dieser Arbeit verwendete Referenzlösung repräsentiert den aktuellen Stand kommerzieller OCR-basierter Systeme. Die Lösung kombiniert spezialisierte OCR-Engines mit KI-basierter Felderkennung („Smart OCR+"), die laut Hersteller eine semantisch orientierte Extraktion relevanter Rechnungsfelder ermöglicht und die manuelle Nachkorrektur reduzieren soll [39].

Insgesamt lässt sich ein allgemeiner Trend zu hybriden Systemen erkennen, die regelbasierte Validierungs- und Plausibilitätsmechanismen mit ML-basierten Extraktionskomponenten kombinieren, wie es etwa Klippa mit DocHorizon bewirbt [40].

Details zu den zugrunde liegenden Verfahren, den genutzten Trainingsdaten sowie den eingesetzten Optimierungs- und Post-Processing-Mechanismen werden von den Anbietern jedoch nicht öffentlich spezifiziert, sodass eine unabhängige Bewertung der Ansätze nur eingeschränkt möglich ist.

4.3 LLM-basierte Ansätze zur Informationsextraktion

Die bisher beschriebenen Verfahren verdeutlichen, dass klassische, regel- und OCR-basierte Ansätze bei der Extraktion von Rechnungsdaten oft an ihre Grenzen stoßen. Large Language Models (LLMs) eröffnen hier neue Perspektiven. Dabei handelt es sich um tiefe neuronale Netze mit teils mehreren hundert Milliarden Parametern, die auf riesigen Mengen natürlicher Sprache trainiert wurden [41]. Ihr zentrales Trainingsziel ist die Vorhersage des nächsten Tokens in einer gegebenen Sequenz („Next-Word Prediction"), wodurch sie in der Lage sind, sprachliche Strukturen, semantische Muster und kontextuelle Abhängigkeiten zu erfassen.

LLMs werden typischerweise in zwei Phasen entwickelt. Im Pre-Training erwerben sie auf einer breiten Textbasis allgemeines Sprachverständnis, im anschließenden Fine-Tuning werden sie für spezifische Anwendungsfälle optimiert. In ihrer Grundform sind sie domänenübergreifend einsetzbar, lassen sich aber durch gezieltes Prompting oder geringe Anpassungen an neue Aufgaben anpassen. Ihre Fähigkeit, auch ohne aufwendiges erneutes Training in unbekannten Aufgaben zu bestehen, wird als Zero-Shot- bzw. Few-Shot-Learning bezeichnet und ist insbesondere für Szenarien mit variablen oder schwer standardisierbaren Eingaben relevant [42].

Die technologische Basis moderner LLMs bilden Decoder-only-Transformer-Modelle. Sie gehen auf die allgemeine Transformer-Architektur [43], und wurden mit der GPT-Reihe erstmals praktisch umgesetzt [44], [45]. Abbildung 4.3 zeigt die grundlegende Architektur eines solchen Modells mit den charakteristischen Komponenten wie Token Embedding, Positional Encoding und dem zentralen Decoder-Block mit Masked Self-Attention.

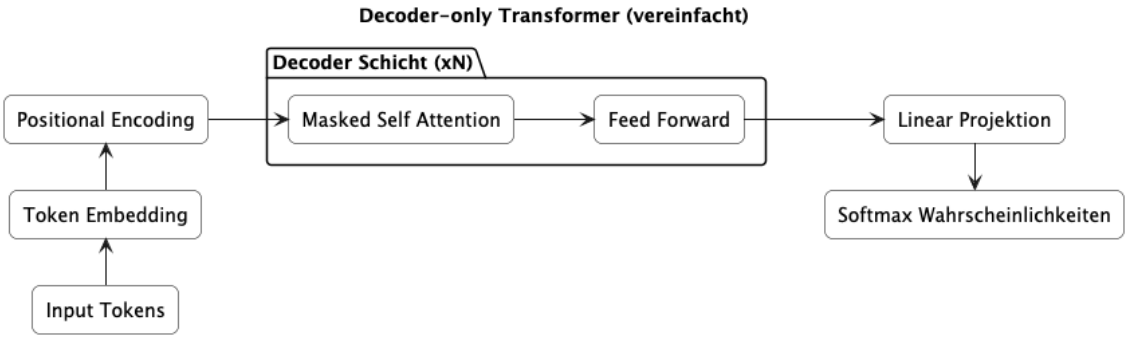


Abbildung 4.3 Vereinfachte Architektur: Decoder-only-Transformers-Model

Diese Architektur ermöglicht es, Texte parallel zu verarbeiten und über Mechanismen wie Self-Attention kontextrelevante Teile einer Eingabe gezielt zu gewichten. Sie hat sich als besonders leistungsfähig für komplexe Sprachverarbeitungsaufgaben erwiesen und bildet die Grundlage für aktuelle Modelle wie GPT-3 [41] oder Claude [46].

LLMs werden zunehmend auch in der strukturierten Dokumentenanalyse eingesetzt, da sie über reines Sprachverständnis hinaus semantische und teilweise auch strukturelle Beziehungen erkennen können. Im Gegensatz zu stark regelbasierten Pipelines kombinieren sie in einem Modell sowohl kontextuelle Interpretation als auch inhaltliche Extraktion, was neue Möglichkeiten für die Verarbeitung komplexer, variantenreicher Rechnungsdokumente eröffnet.

4.3.1 Transformer-Modelle für strukturierte Dokumente

Die Einführung der Transformer-Architektur [43] markierte einen Wendepunkt in der automatisierten Textverarbeitung und legte den Grundstein für moderne Large Language Models. Der revolutionäre „Attention is All You Need"-Ansatz ermöglichte es erstmals, komplexe sprachliche Beziehungen allein auf Basis von Aufmerksamkeitsmechanismen zu erfassen. Im Zentrum steht dabei das sogenannte Attention-Prinzip. Statt Wörter sequenziell zu verarbeiten, wie es bei früheren Modellen der Fall war, analysieren Transformer alle Wörter eines Textes gleichzeitig und gewichten ihre Bedeutung in Abhängigkeit vom Kontext. So kann das Modell etwa erkennen, dass sich das Wort „Rechnung" auf ein Datum oder einen Betrag bezieht, selbst wenn diese Informationen im Text weit voneinander entfernt stehen. Dieser parallele Verarbeitungsansatz ermöglicht ein tiefes Verständnis sprachlicher Zusammenhänge und bildet die Grundlage für die Leistungsfähigkeit moderner LLMs.

BERT (Bidirectional Encoder Representations from Transformers) wurde 2019 vorgestellt und demonstrierte das Potenzial bidirektionaler Sprachmodelle für verschiedene NLP-Aufgaben [47]. Dabei wird betont, dass BERT durch „bidirectional representations" erstmals in der Lage ist, den gesamten umgebenden Kontext eines Tokens gleichzeitig zu berücksichtigen, indem es in allen Schichten sowohl den links- als auch den rechtsseitigen Kontext einbezieht. Dies erwies sich insbesondere bei der Interpretation fragmentierter oder unstrukturierter Textinhalte als vorteilhaft.

LayoutLLM erweiterte das BERT-Konzept um räumliche Koordinateninformationen und ermöglichte dadurch erstmals die Integration von Textinhalt und visueller Position in einem einheitlichen Modell [14]. Diese multimodale Architektur zeigte signifikante Verbesserungen bei dokumentenspezifischen Aufgaben, blieb jedoch auf Dokumente mit verfügbaren Positionsdaten beschränkt.

Das Donut-Modell (Document Understanding Transformer without OCR) stellte einen radikalen Paradigmenwechsel dar, indem es komplett auf vorgelagerte OCR-Verarbeitung verzichtete und Dokumente direkt als Bilddaten verarbeitete [48]. Dieser Ansatz vermeidet OCR-Fehler, erfordert jedoch erhebliche Rechenressourcen und spezialisierte Trainingsdaten.

Diese Entwicklungslinie zeigt den Trend weg von pipeline-basierten Systemen hin zu End-to-End-Ansätzen, die semantisches Verständnis und strukturelle Analyse in einem einheitlichen Modell kombinieren. Moderne LLMs setzen diese Entwicklung fort, indem sie durch reine Sprachmodellierung auch bei vollständig strukturlosen Textinputs zuverlässige Extraktionen ermöglichen.

4.3.2 LLMs für Rechnungsdatenextraktion

Die Anwendung von Large Language Models auf die spezifische Domäne der Rechnungsdatenextraktion ist ein relativ junges Forschungsfeld, das erst mit der Verfügbarkeit leistungsfähiger Modelle wie GPT-3 und dessen Nachfolgern praktische Relevanz erlangt. Aktuelle Studien zeigen vielversprechende Ergebnisse, die das Potenzial LLM-basierter Ansätze zur Überwindung traditioneller Extraktionslimitationen verdeutlichen.

Eine umfassende Untersuchung stellt eine LLM-zentrierte Pipeline zur Informationsextraktion aus Rechnungen vor und vergleicht verschiedene LLM-Ansätze mit traditionellen OCR-basierten Methoden. Die Ergebnisse zeigen, dass moderne LLMs auch bei strukturell fragmentierten Textinputs konsistent bessere Extraktionsraten erreichen als spezialisierte OCR-Systeme und zugleich eine hohe Robustheit gegenüber Layoutvariationen und unkonventionellen Dokumentstrukturen aufweisen [49].

Ein alternativer Ansatz verfolgt ein Zero-Shot-QA-Ensemble (VESPA) und belegt, dass LLMs auch ohne spezifisches Training Rechnungsdaten mit hoher Genauigkeit extrahieren können. Auf realen Rechnungsdatensätzen wurde dabei ein durchschnittlicher F1-Score von 87,5% erzielt, was mehrere etablierte kommerzielle Lösungen übertraf [50].

Darüber hinaus wurde mit ExTTNet ein spezialisiertes Deep-Learning-Modell für tabellenbasierte Extraktionsaufgaben in Rechnungen entwickelt. In Kombination mit einer vorgelagerten OCR-Vorverarbeitung (z. B. Tesseract) erreichte dieses einen F1-Score von bis zu 0,92 und verdeutlicht damit das Potenzial domänenspezifisch optimierter Architekturen, insbesondere für strukturierte Rechnungsbestandteile wie Tabellen [51].

Die in diesen Studien identifizierten Schlüsselfaktoren für erfolgreiche LLM-basierte Extraktion umfassen:

- Zero-Shot-Fähigkeiten: LLMs können ohne vorheriges Training auf spezifische Dokumenttypen angewandt werden und erzielen bereits bei erstmaliger Anwendung brauchbare Ergebnisse.
- **Kontextuelle Interpretation**: Anders als regelbasierte Systeme verstehen LLMs semantische Zusammenhänge und können auch bei fehlenden oder unvollständigen Schlüsselwörtern korrekte Zuordnungen vornehmen.
- **Skalierbarkeit**: Neue Dokumenttypen oder Formate erfordern keine aufwendigen Training-Zyklen, sondern können durch Prompt-Anpassungen abgedeckt werden.

Die aktuellen Forschungsergebnisse verdeutlichen einheitlich, dass LLM-basierte Ansätze das Potenzial haben, traditionelle OCR-Pipelines zu ersetzen oder erheblich zu verbessern, wobei die optimale Implementierung stark von spezifischen Anwendungsanforderungen und verfügbaren Rechenressourcen abhängt.

4.3.3 Zero-Shot- und Few-Shot-Learning

Ein entscheidender Vorteil moderner Large Language Models liegt in ihrer Fähigkeit, Aufgaben ohne vorheriges task-spezifisches Training (Zero-Shot) oder mit minimalem Training anhand weniger Beispiele (Few-Shot) zu lösen. Diese Eigenschaft wurde erstmals systematisch im Kontext von GPT-3 beschrieben und zeigte, dass große vortrainierte Sprachmodelle Aufgaben ohne spezifisches Fine-Tuning bewältigen können. Damit wurde die Grundlage für heutige Zero- und Few-Shot-Strategien geschaffen [41].

Zero-Shot-Learning

Beim Zero-Shot-Learning nutzt das Large Language Model ausschließlich sein während des Pre-Trainings erlerntes Welt- und Sprachwissen, um Aufgaben semantisch zu lösen. Studien zeigen, dass GPT-3 durch rein textuelle Instruktionen in der Lage ist, zuvor ungesehene Aufgaben mit hoher Genauigkeit zu bewältigen [41]. Aktuelle Arbeiten bestätigen diese Ergebnisse für verschiedene Benchmarks und argumentieren, dass Zero-Shot-Prompting insbesondere bei gut standardisierten Aufgaben konkurrenzfähige Ergebnisse zu spezialisierten Modellen liefern kann [52].

Für den Rechnungsdatenkontext legt dies nahe, dass LLMs Beträge, Datumsangaben oder Rechnungsnummern ohne explizites Training extrahieren können, indem sie auf erlernte Muster und allgemeines semantisches Wissen zurückgreifen.

Few-Shot-Learning

Few-Shot-Learning erweitert den Zero-Shot-Ansatz, indem das Modell eine kleine Anzahl von Beispielaufgaben innerhalb des Prompts (In-Context Learning) erhält. Es „lernt" dabei nicht im trainingsbezogenen Sinn, sondern nutzt seine bereits gelernten Muster, um die gezeigten Beispiele zu imitieren und ähnliche Eingaben konsistent zu beantworten [41]. Dadurch kann das Modell spezifische Formatierungsanforderungen, domänenspezifische Muster und in begrenztem Maße auch seltene Varianten berücksichtigen. Typische Anwendungsfälle sind:

- **Formatvorgaben** (z. B. gewünschte JSON-Struktur)
- **Edge-Case-Handling** (z. B. unkonventionelle Schreibweisen oder seltene Layoutvarianten)
- **Domänenspezifische Terminologie** (z. B. branchenspezifische Begriffe)

Untersuchungen zeigen zudem, dass die Few-Shot-Leistung stark von der Prompt-Kalibrierung abhängt. Eine gezielte Anpassung der Beispielreihenfolge, Formulierungen und Label-Verteilung steigert die Zuverlässigkeit insbesondere bei Edge-Cases und domänenspezifischen Aufgaben signifikant [53].

Chain-of-Thought Prompting

Chain-of-Thought (CoT) Prompting erweitert Zero-Shot- und Few-Shot-Ansätze, indem das Modell seinen Reasoning-Prozess explizit durchläuft, anstatt direkt ein Endergebnis zu liefern.

- **Zero-Shot CoT**: Das Modell wird mit einer Anweisung wie „Think step by step" oder „Let's think step by step" zu schrittweisem Reasoning aufgefordert, ohne Beispiele [52].
- **Few-Shot CoT**: Beispiele enthalten explizite Zwischenschritte, die das Modell imitiert [54].

Eine praxisrelevante Abwandlung stellt das sogenannte Hidden CoT dar, bei dem das Modell intern schrittweise denkt, aber nur das finale Ergebnis ausgibt. Erste Untersuchungen zeigen, dass Hidden CoT eine effektive Strategie für strukturierte Extraktionsaufgaben darstellt, da es den Aufwand für nachgelagerte Post-Processing-Schritte reduziert, eine hohe Konsistenz der Ergebnisse ermöglicht und zugleich die generierten Ausgabetoken sowie damit verbundene Kosten verringert [55].

Das folgende Beispiel illustriert einen möglichen Aufbau eines Hidden-CoT-Prompts, wie er zur Extraktion von Rechnungsdaten gestaltet sein könnte.

```
1. Finde alle Kandidaten für Rechnungsnummer, Datum und Beträge
2. Prüfe das Format:
        • Rechnungsnummer: alphanumerisch.
        • Datum: ISO-Format (YYYY-MM-DD).
        • Betrag: Double, Punkt als Dezimaltrennzeichen, kein Tausendertrennzeichen.
3. Wähle den plausibelsten Betrag
4. Gib nur das Ergebnis als JSON zurück, ohne Erklärungen.
```

Abbildung 4.4 Chain-of-Thought Beispielprompt

Studien belegen, dass Chain-of-Thought Prompting insbesondere bei komplexen Dokumentstrukturen die Extraktionsgenauigkeit signifikant verbessert, da es das Modell zu einer systematischeren Analyse zwingt [52], [54].

4.3.4 Herausforderungen und offene Forschungsfragen

Trotz der vielversprechenden Theorien und ersten empirischen Erkenntnisse bestehen bei der praktischen Anwendung von LLMs auf strukturierte Datenextraktion noch erhebliche Herausforderungen, die sowohl methodische als auch technische Aspekte umfassen.

Konsistenz und Halluzinationen

Ein zentrales Problem ist die Konsistenz der LLM-Ausgaben. Selbst bei deterministischen Einstellungen (z. B. temperature = 0) lassen sich gelegentlich Variationen in den Extraktionsergebnissen beobachten, was die Reproduzierbarkeit wissenschaftlicher Untersuchungen beeinträchtigt. Hinzu kommt das Risiko sogenannter Halluzinationen, also der Generierung plausibel erscheinender, jedoch im Originaldokument nicht enthaltener Inhalte. Gerade bei Rechnungen kann dies bedeuten, dass Beträge, Datumsangaben oder Referenznummern erfunden oder fehlerhaft ergänzt werden [56].

Parsing-abhängige Fehlerquellen

Die Qualität der vorgelagerten Textextraktion aus PDF-Dokumenten hat direkten Einfluss auf die LLM-Performance. Fehlerquellen wie unvollständige Texterfassung, fehlerhafte Zeichenkodierung oder das Aufbrechen tabellarischer Strukturen (Abschnitt 4.1) wirken sich unmittelbar auf die Extraktion aus. Vision-unterstützte Ansätze, die Layout- und Bildinformationen direkt in die Modellverarbeitung einbeziehen, können hier Verbesserungen bieten, erhöhen aber die Komplexität und den Rechenaufwand.

Kosten und Laufzeit versus Genauigkeit

Die Balance zwischen Extraktionsgenauigkeit, Verarbeitungsgeschwindigkeit und entstehenden Kosten stellt eine praktische Optimierungsaufgabe dar. Große Cloud-Modelle erreichen in der Regel die höchste Genauigkeit, können jedoch bei massenhaft verarbeiteten Dokumenten durch Kosten und Latenzzeiten prohibitiv sein. Kleinere, lokal betreibbare Modelle bieten eine potenziell kosteneffiziente Alternative, erfordern jedoch Abstriche bei Genauigkeit und Robustheit [57].

4.4 APIs und Integrationsarchitektur

Die praktische Evaluation verschiedener LLM-Ansätze zur Rechnungsdatenextraktion erfordert eine differenzierte Betrachtung der zugrunde liegenden Bereitstellungsmodelle und deren technischen Charakteristika. Die Wahl zwischen Cloud-basierten Services und lokal betreibbaren Open-Source-Alternativen beeinflusst nicht nur die methodische Durchführbarkeit, sondern auch die Reproduzierbarkeit wissenschaftlicher Untersuchungen.

Kommerzielle Cloud-Angebote stellen in der Regel die leistungsstärksten Modelle bereit, gehen jedoch mit Einschränkungen hinsichtlich Kostenkontrolle, Datenhoheit und Transparenz einher. Lokale Open-Source-Modelle ermöglichen hingegen volle Kontrolle über Daten und Evaluationsumgebung, erfordern aber beträchtliche Rechenressourcen und sind in ihrer Leistungsfähigkeit häufig limitiert.

Von besonderer Bedeutung für Vergleichsstudien ist die zunehmende Standardisierung der API-Schnittstellen. Einheitliche Formate, wie sie von Anbietern wie OpenAI und Anthropic oder in Open-Source-Implementierungen wie llama.cpp genutzt werden, erleichtern den Austausch von Modellen und ermöglichen methodische Konsistenz.

Die folgenden Unterabschnitte analysieren die Charakteristika der jeweiligen Bereitstellungsmodelle und ihre Relevanz für wissenschaftliche Evaluationsszenarien.

4.4.1 Geschlossene Modelle (proprietäre Modelle)

Kommerzielle LLM-Anbieter wie OpenAI und Anthropic stellen über Cloud-APIs den Zugang zu hochperformanten, geschlossenen Modellen bereit, die den aktuellen Stand der Technik in vielen NLP-Aufgaben repräsentieren. Diese proprietären Systeme zeichnen sich durch ihre Leistungsfähigkeit und Stabilität aus, bringen jedoch Einschränkungen hinsichtlich Transparenz und Datenschutz mit sich.

OpenAI GPT-4 und GPT-4o gelten als Referenzmodelle für komplexe Textverständnisaufgaben und zeigen konsistent hohe Leistungen bei strukturierten Extraktionsaufgaben [58]. Die Modelle sind für Chat-basierte Interaktionen optimiert und unterstützen sowohl reine Textverarbeitung als auch multimodale Eingaben. GPT-4o erweitert die Grundfunktionalität um eine verbesserte Verarbeitungsgeschwindigkeit und eine höhere Kosteneffizienz bei vergleichbarer Genauigkeit.

Anthropic Claude basiert auf dem Konzept der Constitutional AI, das auf integrierte Sicherheitsmechanismen und erhöhte Transparenz abzielt [59]. Für strukturierte Extraktionsaufgaben zeigt Claude besondere Stärken bei der Einhaltung präziser Formatvorgaben und der Konsistenz der Ausgaben.

Diese geschlossenen Modelle sind für Forschungsanwendungen relevant, da sie ohne lokale Hardware-Infrastruktur verfügbar sind und hohe Genauigkeit bei komplexen Extraktionsaufgaben ermöglichen. API-Parameter wie $temperature = 0$ für deterministische Ausgaben und max_tokens zur Kostenkontrolle sind dabei zentral für reproduzierbare Experimente. Die Konfiguration dieser Parameter hat direkten Einfluss sowohl auf die Extraktionsgenauigkeit als auch auf die entstehenden Kosten pro verarbeitetem Dokument.

4.4.2 Offene Modelle

Open-Source-Sprachmodelle stellen eine praxisnahe Alternative zu geschlossenen Cloud-Diensten dar und lassen sich lokal über Inferenzumgebungen wie LM Studio [60] oder Ollama [61] betreiben. Daraus ergeben sich Vorteile wie volle Datenkontrolle, transparente Kosten durch einmalige Hardware-Investitionen sowie die Möglichkeit zur Anpassung der Modelle. Für wissenschaftliche Evaluationen sind offene Modelle besonders wertvoll, da sie Einblick in Architektur, Trainingsverfahren und Inferenz erlauben und damit die Reproduzierbarkeit unterstützen.

Im Zentrum dieser Arbeit steht Gemma 3, eine von Google Research veröffentlichte Open-Source-Modellreihe [62]. Sie ist in verschiedenen Parametergrößen (1B, 2B, 4B, 12B, 27B) verfügbar, wurde für effiziente Inferenz optimiert und zeichnet sich durch vergleichsweise moderate Hardwareanforderungen aus. Gemma 3 bietet erweiterte Kontextlängen, optimierte Attention-Mechanismen und multimodale Fähigkeiten (Text- und Bildverarbeitung), wodurch es sich besonders für lokale Experimente und wissenschaftliche Untersuchungen eignet.

Darüber hinaus existieren weitere bedeutende offene Modellfamilien wie Llama [63] von Meta AI oder Mistral [64], die die Entwicklung zahlreicher spezialisierter Varianten angestoßen haben und in Forschung wie Praxis eine zentrale Rolle spielen. Über Plattformen wie Hugging Face steht inzwischen eine große Vielfalt an offenen Modellen zur Verfügung, die je nach Anwendungsszenario unterschiedliche Stärken und Kompromisse zwischen Genauigkeit, Geschwindigkeit und Ressourcenverbrauch bieten.

4.4.3 Einheitliche API-Nutzung

Ein wesentlicher technischer Fortschritt in der praktischen LLM-Integration liegt in der Standardisierung der API-Schnittstellen. Nahezu alle relevanten Anbieter bilden ein OpenAI-kompatibles API-Format ab, wodurch eine einheitliche Integration verschiedener Modelle ermöglicht wird.

Die Standardisierung erfolgt primär über den /v1/chat/completions Endpunkt, der eine konsistente Schnittstelle für Chat-basierte Interaktionen bereitstellt. Dieser Ansatz ermöglicht es, verschiedene Modelle mit identischen Client-Implementierungen anzusprechen, was für Vergleichsstudien von zentraler Bedeutung ist.

Wesentliche in dieser Arbeit verwendete API-Parameter umfassen:

- **model**: Spezifikation des zu verwendenden Modells (z.B. „gpt-4.1", „claude-3-sonnet", „gemma-3-4b-it")
- **messages**: Liste von Nachrichten mit Rollenzuweisung (System, User, Assistent), die das Verhalten des Modells steuert.
- **temperature**: Der Temperature-Parameter steuert die Ausgabevariabilität (Randomness) eines Sprachmodells. Ein niedriger Wert (nahe 0) führt zu deterministischeren Ausgaben. Höhere Werte fördern teils kreativere Antworten, gehen jedoch mit einer erhöhten Wahrscheinlichkeit inkonsistenter Ausgaben einher. Der sinnvolle Wertebereich variiert modellabhängig, liegt jedoch typischerweise zwischen 0 und 2.
- **top_p**: legt fest, bis zu welcher Wahrscheinlichkeit Tokens berücksichtigt werden (0 für deterministische Aufgaben)
- **max_tokens**: Begrenzung der Antwortlänge zur Kostenkontrolle

Für strukturierte Extraktionsaufgaben empfiehlt sich die Verwendung von $temperature = 0$, da sie die Variabilität der Ausgaben minimiert und damit konsistentere Ergebnisse ermöglicht [65]. Der max_tokens Parameter wird typischerweise konservativ gewählt, um sowohl Kosten zu begrenzen als auch sicherzustellen, dass strukturierte JSON-Ausgaben vollständig übertragen werden.

Diese API-Standardisierung ist für wissenschaftliche Untersuchungen von fundamentaler Bedeutung, da sie ermöglicht, Extraktionsstrategien anbieterübergreifend zu implementieren und zu reproduzieren. Experimente können dadurch verschiedene Modelle mit identischen Prompts und Parametern vergleichen, ohne modellspezifische Implementierungsunterschiede berücksichtigen zu müssen.

5 Versuchsaufbau und Methodik

Die Untersuchung der Leistungsfähigkeit verschiedener Large Language Models (LLMs) bei der Rechnungsdatenextraktion im Vergleich zu dem von aifinyo eingesetzten Rechnungs-OCR (Gini) erfordert einen Versuchsaufbau, der sowohl die spezifischen Anforderungen des Unternehmens als auch die aktuellen Möglichkeiten der LLMs berücksichtigt. Verschiedene Extraktionsstrategien werden dabei auf identischen Datensätzen getestet und verglichen, um ein umfassendes Bild der jeweiligen Vor- und Nachteile zu erhalten und Aussagen über Optimierungspotenziale treffen zu können.

Grundlage der Versuchsreihe ist eine eigens entwickelte Evaluationsplattform. Sie ermöglicht es, verschiedene Extraktionsstrategien anzulegen, diese gegen definierte Sample-Sets auszuführen und die Ergebnisse automatisch zu vergleichen. Ferner erzeugt sie für den Vergleich notwendige KPIs und persistiert alle Ergebnisse konsistent. Auf diese Weise stellt sie eine wiederholbare und gründlich dokumentierte Evaluation der unterschiedlichen Ansätze sicher.

Die Plattform ist dabei nicht nur ein technisches Hilfsmittel, sondern auch ein Forschungsartefakt, das im Sinne der gestaltungsorientierten Wirtschaftsinformatik entwickelt wird [66]. Darüber hinaus orientiert sich der Entwurf am Konzept von „Experimentation Workbenches", wie es in der Softwaretechnik vorgeschlagen wurde [67]. Dieses Konzept betont die Bedeutung von systematischer Unterstützung für Reproduzierbarkeit, Wiederholbarkeit und Nachvollziehbarkeit, die in der Plattform konsequent umgesetzt sind. Der Entwurf folgt somit dem Prinzip, Artefakte so zu gestalten, dass sie sowohl wissenschaftliche Strenge (Rigor) als auch praktische Relevanz (Relevance) sicherstellen.

Im Folgenden werden der experimentelle Aufbau, die verwendeten Daten, die technische Infrastruktur sowie das Evaluationskonzept und die angewandten Metriken beschrieben.

5.1 Experimenteller Aufbau und praktische Durchführung

Der experimentelle Aufbau basiert auf einem dokumentbasierten Vergleichsansatz. Jedes einzelne Rechnungsdokument wird dabei durch mehrere LLM-Strategien verarbeitet und mit den historischen Extraktionsergebnissen des bestehenden Rechnungs-OCRs sowie den anderen Strategien verglichen. Die Evaluation erfolgt in zwei Phasen. Zu Beginn wird eine iterative Optimierung anhand eines speziell zusammengestellten, inhaltlich komplexen Developmentdatensatzes mit 3.000 Dokumenten durchgeführt. Anschließend erfolgt eine finale Blind-Evaluation auf einem unabhängigen, repräsentativen Evaluationsdatensatz mit 10.000 Dokumenten. Dieser wird zufällig aus dem Gesamtdatenbestand von über 700.000 Rechnungen erstellt, wobei Überschneidungen zwischen den beiden Datensätzen verhindert werden.

Die Infact-Referenzdaten, die durch manuelle Überprüfung und Korrektur der Gini-OCR-Extraktion entstanden sind, dienen als Ground Truth für die Bewertung. Gegen diese Daten werden sowohl die historischen Gini-Ergebnisse als auch die in Abschnitt 4.3 beschriebenen LLM-Ansätze verglichen.

Im Fokus der Evaluation stehen drei zentrale Datenfelder, die für die Rechnungsverarbeitung bei aifinyo von hoher Bedeutung sind. Dabei handelt es sich um die Rechnungsnummer als eindeutige Identifikation, den Rechnungsbetrag sowie das Rechnungsdatum. Diese Felder werden bewusst ausgewählt, da sie sowohl für die automatisierte Verarbeitung als auch für die manuelle Nachprüfung klar abgrenzbar und validierbar sind. Zudem weisen sie aufgrund ihrer Geschäftskritikalität die höchste Datenqualität in den Infact-Referenzdaten auf. Andere Felder bleiben unberücksichtigt, um den Fokus auf die robustesten und vergleichbarsten Kernfelder zu legen.

Die praktische Durchführung der Experimente erfolgt in mehreren Schritten. Zunächst werden alle Modelle mit einem einfachen Zero-Shot-Prompt in unterschiedlichen Konfigurationen getestet, um eine erste Vergleichsbasis zu schaffen und potenzielle Fehlerquellen in den Daten zu identifizieren. Nach den Korrekturen der Referenzdaten und den ersten Erkenntnissen kommen optimierte Zero-Shot-Prompts sowie Few-Shot-Prompts zum Einsatz, um leistungsfähige Strategien für die finale Evaluation zu identifizieren. Parallel dazu erfolgt ein Benchmarking mehrerer Textextraktionsbibliotheken (Abschnitt 6.3.3), um eine robuste und konsistente Basis für die LLM-Verarbeitung sicherzustellen.

Dieses Vorgehen ermöglicht eine schrittweise Verfeinerung der Strategien und liefert eine fundierte Grundlage für die abschließende Bewertung anhand des Evaluationsdatensatzes.

5.2 Datengrundlage der Evaluationsplattform

Die in dieser Arbeit durchgeführten Versuche greifen auf den umfangreichen Dokumentenbestand der unternehmensinternen Factoring-Software Infact zurück, der eine breite Vielfalt an Layouts, Kreditoren und Branchen abdeckt. In mehr als sechs Jahren Geschäftstätigkeit hat sich ein Gesamtbestand von über 700.000 Rechnungsdokumenten akkumuliert, von denen etwa 400.000 hochgeladen wurden und damit für die Evaluation zur Verfügung stehen. Ausschließlich diese Rechnungen werden berücksichtigt, da sie eine automatisierte Extraktion erfordern. Die restlichen vom System selbst generierten Rechnungen, deren Daten bereits in strukturierter Form vorliegen, sowie eingereichte Scans in reiner Bildform werden ausgeschlossen. Letztere stellen zwar grundsätzlich eine Extraktionsherausforderung dar, fallen jedoch nicht in den Fokus dieser Arbeit, da sie zunächst einer vorgelagerten OCR-Verarbeitung bedürfen.

Die für die Experimente verwendeten Referenzdaten beruhen auf den ursprünglich durch das Rechnungs-OCR extrahierten Werten, die im Rahmen der operativen Prozesse manuell überprüft und korrigiert wurden. Auf diese Weise entstanden die sogenannten Infact-Referenzdaten, die die derzeit verlässlichste verfügbare Basis für die Bewertung der LLM-Extraktionen darstellen. Dennoch bleibt ein Restrisiko vereinzelter Fehler bestehen, da auch manuelle Kontrollen erfahrungsgemäß fehleranfällig sind [68].

Die OCR- und Infact-Referenzdaten werden in strukturierter Form persistiert und bilden die zentrale Grundlage für den Vergleich der verschiedenen Extraktionsstrategien. Die genaue technische Umsetzung der Evaluation erfolgt über eine eigens entwickelte Evaluationsplattform, deren Aufbau und Funktionsweise in Abschnitt 6 beschrieben werden.

5.3 Evaluationskonzept und Metriken

Die im vorangegangenen Abschnitt beschriebene Evaluationsplattform bildet die Grundlage für die Bewertung der verschiedenen LLM-basierten Extraktionsstrategien. Ziel der Evaluation ist es, die Leistungsfähigkeit der Modelle unter realistischen Bedingungen systematisch zu messen und vergleichbar zu machen. Neben der reinen Extraktionsgenauigkeit wird dabei auch die wirtschaftliche Effizienz berücksichtigt, da sowohl Tokenverbrauch als auch Verarbeitungsdauer in der Plattform erfasst werden.

Die Bewertung erfolgt feldbasiert auf Dokumentenebene und nutzt die in der Plattform hinterlegten Infact-Referenzdaten als Ground Truth. Im Fokus stehen Rechnungsdokumente, deren fehlerfreie Datenerfassung eine zentrale Voraussetzung für nachgelagerte Prozesse wie Zahlungsfreigaben, Mahnwesen oder Limitprüfungen darstellt. Eine unzureichende Extraktionsqualität kann in diesen Bereichen unmittelbar zu Fehlentscheidungen führen, weshalb eine präzise und nachvollziehbare Bewertung essenziell ist.

Zur Absicherung der Ergebnisse wird ergänzend eine Signifikanzprüfung durchgeführt. Dabei wird insbesondere mit dem McNemar-Test untersucht, ob Unterschiede zwischen Strategien statistisch belastbar sind. Ergänzende Kennzahlen wie Odds-Ratio und Wilson-Konfidenzintervalle ermöglichen eine differenziertere Interpretation und helfen, zufällige Befunde von echten Leistungsunterschieden zu trennen.

5.3.1 Zielsetzung der Evaluation

Die Evaluation verfolgt das Ziel, die Extraktionsleistung der verschiedenen LLM-basierten Strategien objektiv und nachvollziehbar zu bewerten. Der Fokus liegt auf einer feldbasierten Analyse, da schon ein falsch extrahierter Wert unmittelbare Auswirkungen auf geschäftskritische Prozesse haben kann. Besonders relevant sind die Felder Rechnungsnummer, Rechnungsdatum und Rechnungsbetrag, da sie für Zahlungsfreigaben, Mahnwesen und Limitprüfungen essenziell sind. Die Bewertung erfolgt dokumentenbasiert und vergleicht die von den Modellen extrahierten Werte mit den in den Infact-Referenzdaten hinterlegten Ground-Truth-Werten.

5.3.2 Metriken und Bewertungslogik

Zur Bewertung der Extraktionsqualität wird die Genauigkeit als primäre Metrik herangezogen. Sie ermöglicht sowohl eine feldspezifische als auch eine feldübergreifende Analyse der Ergebnisse. Die Berechnung erfolgt feldweise innerhalb eines Dokuments, um Unterschiede in der Erkennungsleistung für einzelne Rechnungsfelder sichtbar zu machen.

Ein Feldwert gilt als korrekt, wenn er exakt oder nach einer definierten Normalisierung mit dem Ground-Truth-Wert übereinstimmt. Fehlende oder abweichende Werte werden als inkorrekt gewertet. Bei der hier betrachteten Rechnungsdatenextraktion mit fest definierten, obligatorischen Feldern entfällt im Unterschied zu binären Klassifikationsaufgaben wie beispielsweise der Spam-Erkennung die Kategorie der True Negatives, da jedes Feld einen konkreten Sollwert besitzt und das Ergebnis ausschließlich korrekt oder inkorrekt sein kann. Damit wird die Genauigkeit in einer auf die Rechnungsdatenextraktion angepassten Form berechnet, die sich an der klassischen Definition [69] orientiert, nach der sie als Verhältnis korrekt klassifizierter Instanzen zur Gesamtzahl definiert ist.

Um konsistente Vergleiche zu ermöglichen, erfolgt eine feldspezifische Normalisierung der extrahierten Daten nach den in Tabelle 5.1 definierten Regeln.

Feld	**Normalisierung**
Datum	Konvertierung in das ISO-Format YYYY-MM-DD.
Betrag	Vergleich als numerische Float-Werte.
Rechnungsnummer	Entfernung aller Leerzeichen sowie Umwandlung von Gedankenstrichen (–) in Bindestriche (-), um formatbedingte Unterschiede, etwa zwischen „RE–2024/001" und „RE-2024 / 001", zu vermeiden.

Tabelle 5.1 Normalisierungen

Neben der feldspezifischen Betrachtung werden die Ergebnisse auch in aggregierter Form bewertet. Ergänzend zur $Overall\ Accuracy$ wird eine $Document\ Accuracy$ ermittelt, bei der ein Dokument nur dann als korrekt gilt, wenn alle drei geprüften Felder (Rechnungsnummer, Datum, Betrag) fehlerfrei extrahiert wurden. Diese strenge Metrik spiegelt den realen Business-Impact wider, da bereits ein einzelner Fehler zu nachgelagerten Prozessproblemen führen kann.

Zur Sicherstellung einer einheitlichen Auswertung sind alle verwendeten Metriken wie folgt definiert:

Overall Accuracy: Verhältnis korrekt extrahierter Felder zur Gesamtzahl geprüfter Felder, liefert einen Überblick über die generelle Extraktionsqualität.

$$\text{Overall Accuracy} = \frac{\text{Anzahl korrekt extrahierter Felder über alle Dokumente}}{\text{Gesamtzahl geprüfter Felder über alle Dokumente}}$$

Document Accuracy: strenge Metrik, bei der ein Dokument nur dann als korrekt gilt, wenn alle Felder fehlerfrei extrahiert wurden.

$$\text{Document Accuracy} = \frac{\text{Anzahl Dokumente mit allen Feldern korrekt}}{\text{Gesamtzahl geprüfter Dokumente}}$$

Number, Date und Amount Accuracy: Feldspezifische Genauigkeiten, die getrennt für Rechnungsnummer, Rechnungsdatum und Rechnungsbetrag ausgewiesen werden.

$$\text{Number Accuracy} = \frac{\text{Anzahl Dokumente mit korrekter Rechnungsnummer}}{\text{Gesamtzahl geprüfter Dokumente}}$$

$$\text{Date Accuracy} = \frac{\text{Anzahl Dokumente mit korrektem Rechnungsdatum}}{\text{Gesamtzahl geprüfter Dokumente}}$$

$$\text{Amount Accuracy} = \frac{\text{Anzahl Dokumente mit korrektem Rechnungsbetrag}}{\text{Gesamtzahl geprüfter Dokumente}}$$

5.3.3 Tokenverbrauch / Laufzeit

Neben der reinen Extraktionsqualität werden auch Tokenverbrauch und Verarbeitungsdauer je Dokument erfasst und in der Evaluationsplattform ausgewertet. Diese Kennzahlen dienen der ergänzenden Beurteilung von Wirtschaftlichkeit und Effizienz der Strategien, stehen jedoch nicht im Mittelpunkt dieser Arbeit. Eine direkte ökonomische Bewertung, beispielsweise in Form einer Abwägung zusätzlicher Kosten oder Laufzeit pro Prozentpunkt höherer Genauigkeit, erfolgt nicht.

5.3.4 Signifikanzprüfung

Zur Bewertung der Unterschiede zwischen den getesteten Extraktionsstrategien wird der McNemar-Test verwendet [70]. Er eignet sich für gepaarte Daten und betrachtet nur die Fälle, in denen sich die Systeme unterscheiden, also wenn ein Modell korrekt und das andere falsch liegt. Der Test wird sowohl für Vergleiche mit der OCR-Baseline (Gini) als auch für direkte Vergleiche zwischen den LLM-Strategien eingesetzt. Neben dem $p - Wert$ werden das Odds-Ratio (b/c) mit 95-%-Konfidenzintervall sowie der zugehörige χ^2-Wert angegeben [71]. Um zufällige Befunde bei mehreren Tests zu vermeiden, wird die Holm-Bonferroni-Korrektur angewendet [72].

Die Signifikanzprüfung erfolgt nur auf Dokumentebene. Für die $Overall\ Accuracy$ wird kein Test berechnet, da die Feldinstanzen eines Dokuments voneinander abhängen und somit keine unabhängige Stichprobe darstellen. Zur Darstellung der Unsicherheit werden Wilson-Konfidenzintervalle der $Document\ Accuracy$ angegeben [73], [74]. Diese Methode ist gegenüber der klassischen Normalapproximation robuster, insbesondere wenn die beobachtete $Accuracy$ sehr hoch oder sehr niedrig ist und die resultierende Verteilung entsprechend asymmetrisch wird. Das 95-Prozent-Intervall berechnet sich nach der Wilson-Formel.

$$CI = \frac{\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

Formel 5.1 Wilson-Formel

Hierbei bezeichnet $\hat{p}$ den beobachteten Anteil, n die Stichprobengröße und z das entsprechende Quantil der Standardnormalverteilung (für 95% gilt $z = 1{,}96$). Ein Ergebnis gilt als statistisch signifikant, wenn der $p - Wert$ nach Holm-Korrektur kleiner als 0,05 ist.

5.4 Stichprobenstrategie

Die Auswahl und Zusammenstellung der Datensätze folgt einer zweistufigen Stichprobenstrategie, die sowohl die iterative Optimierung der Extraktionsstrategien als auch eine belastbare, repräsentative Endbewertung sicherstellen soll.

Dieses Vorgehen orientiert sich an etablierten Prinzipien der Machine-Learning-Evaluation, bei denen ein Trainings- bzw. Entwicklungsdatensatz von einem unabhängigen Evaluationsdatensatz getrennt wird, um Überoptimierung zu vermeiden und eine belastbare Aussagekraft zu sichern [75].

Darüber hinaus wird die Konstruktion des Developmentsets bewusst fehlerfokussiert angelegt. Ähnlich wie im „Hard Sample Aware Prompt-Tuning" [76] werden gezielt schwierige Beispiele berücksichtigt, um die Optimierungspotenziale der Strategien besser sichtbar zu machen. Während im Originalansatz die Auswahl problematischer Instanzen über Reinforcement Learning erfolgt, setzt das hier verwendete Verfahren auf ein regelbasiertes Sampling. Dies gewährleistet eine hohe Layoutdiversität und eine breite Abdeckung extraktionskritischer Merkmale.

Developmentdatensatz

Für die Entwicklung und Optimierung der Strategien wird ein gezielt zusammengestellter Developmentdatensatz mit 3000 Dokumenten verwendet. Dieser Datensatz enthält bewusst einen überproportional hohen Anteil komplexer und fehleranfälliger Fälle. Hierzu zählen insbesondere Rechnungen mit Diskrepanzen zwischen den ursprünglichen OCR-Ergebnissen und den manuell korrigierten Referenzdaten sowie eine hohe Diversität an Kreditoren mit unterschiedlichen Layouts. Ziel ist es, Schwächen der Modelle frühzeitig sichtbar zu machen und gezielt Optimierungspotenziale zu identifizieren.

Die Größe von 3000 Dokumenten stellt einen pragmatischen Kompromiss dar, groß genug, um statistisch robuste Aussagen zu ermöglichen, und gleichzeitig klein genug, um im Entwicklungsprozess effizient eingesetzt werden zu können. Die Präzision lässt sich über die Statistik der Anteilsmaße begründen (vgl. Abschnitt 5.3.4). Die Standardabweichung eines Anteils $\hat{p}$ ergibt sich zu $\sqrt{\hat{p}(1-\hat{p})/n}$. Bei einer Stichprobe von $n = 3000$ Dokumenten liegt sie selbst für moderate Genauigkeiten von ($\hat{p} \approx 0{,}95$) Prozent bei rund 0,4 Prozentpunkten. Unter Anwendung der Wilson-Formel für Konfidenzintervalle (vgl. Abschnitt 5.3.4) verengt sich die Unsicherheit damit auf etwa ±0,5 Prozentpunkte. Für sehr hohe Genauigkeiten in der Nähe von 99% sinkt sie auf rund ±0,3 Prozentpunkte. Damit lassen sich auch kleine Unterschiede zwischen Strategien verlässlich erkennen.

Evaluationsdatensatz

Die finale Bewertung erfolgt auf einem unabhängigen Evaluationsdatensatz mit 10.000 Dokumenten, der zufällig aus dem Gesamtdatenbestand von etwa 400.000 Rechnungen gezogen wird. Dieser Datensatz repräsentiert den realen Einsatzkontext und dient als Grundlage für die abschließende praxisnahe Bewertung der Strategien. Durch die größere Stichprobengröße reduziert sich die statistische Unsicherheit nochmals erheblich. Bei 10.000 Dokumenten liegen die Wilson-Intervalle selbst bei Genauigkeiten um 95% im Bereich von nur ±0,3 Prozentpunkten, bei höheren Anteilen sogar darunter. Auf diese Weise können auch sehr kleine Leistungsunterschiede zwischen Strategien zuverlässig nachgewiesen werden.

Vermeidung von Überschneidungen

Die beiden Datensätze sind strikt voneinander getrennt. Dokumente, die im Developmentdatensatz enthalten sind, werden aus dem Evaluationsdatensatz ausgeschlossen. Dadurch wird verhindert, dass Strategien implizit an bereits bekannten Beispielen bewertet werden, und die finale Evaluation behält ihren Blind-Test-Charakter.

5.5 Evaluationsumgebung

Für die Durchführung der Experimente kommen sowohl proprietäre als auch Open-Source-Modelle zum Einsatz. Die technische Infrastruktur sowie alle Parameter werden so gewählt, dass eine hohe Reproduzierbarkeit und Vergleichbarkeit gewährleistet sind.

Für den Vergleich wurden in dieser Arbeit folgende Modelle verwendet:

Claude 3 Sonnet (20250219) [77]

Das Modell wird über die Infrastruktur von Anthropic ausgeführt. Da keine deterministische Seed-Steuerung möglich ist, erfolgen alle Aufrufe, sofern nicht anders angegeben, mit den Standardparametern $temperature = 0$ und $top_p = 1$, um eine möglichst hohe Konsistenz in den Ergebnissen zu gewährleisten.

GPT-4.1 [78]

Die Ausführung erfolgt über die OpenAI-API. Zur Sicherstellung reproduzierbarer Ergebnisse wird ein fixer Seed (3333) verwendet. Die weiteren Modellparameter sind ebenfalls standardisiert: $temperature = 0$, $top\_p = 1$.

Gemma 3-IT (1B - 27B) [62]

Die Open-Source-Modelle werden lokal über LMStudio [60] im GGUF-Format betrieben, das als Standardformat für quantisierte LLMs in llama.cpp-basierten Umgebungen etabliert ist [79]. Für alle Experimente wurden identische Parameter verwendet. Als Seed wird immer 3333 verwendet. Die Ausführung erfolgt auf der nachfolgend aufgeführten Hardware.

Hardware

Die Ausführung der Open-Source-Modelle erfolgt auf einer lokalen Workstation mit folgender Ausstattung:

- **GPU**: NVIDIA GeForce RTX 4090
- **CPU**: Intel Core i9-12900K (12th Gen)
- **RAM**: 4× Corsair Vengeance DDR5, 32 GB, 6400 MHz, CL32
  - **Software**: LMStudio (v. 0.3.17) + CUDA llama.cpp (v. 1.42.0) für GGUF-Modelle
  - **LMStudio**: Es wurden die Basiskonfigurationen verwendet. Lediglich die Kontextlänge wurde auf 8192 eingestellt und es wurde ein fester $seed = 3333$ definiert.

Diese Umgebung erlaubt es, auch große Modelle (z. B. Gemma 27B-IT) effizient auszuführen und sämtliche Tests lokal unter kontrollierten Bedingungen zu wiederholen. Proprietäre Modelle (GPT, Claude) werden über standardisierte API-Schnittstellen angebunden. Eine zentrale Logging- und Vergleichskomponente innerhalb der Evaluationsplattform gewährleistet eine einheitliche Ergebnisstruktur.

5.6 Experimenteller Ablauf

Der experimentelle Ablauf erfolgt vollständig innerhalb der im Rahmen dieser Arbeit entwickelten Evaluationsplattform. Diese ist nicht nur ein statisches Werkzeug, sondern wird während der Durchführung iterativ erweitert und an neue Anforderungen angepasst. Die Plattform entwickelt sich evolutionär mit den im Verlauf der Evaluation gewonnenen Erkenntnissen weiter.

Neue Funktionen, wie zusätzliche Metriken oder Visualisierungen, entstehen bedarfsorientiert, um die Analyse zu verfeinern.

Die Durchführung folgt einer klar strukturierten Abfolge, die eine systematische und reproduzierbare Evaluation sicherstellt:

1. **Initiale Tests (Baseline)**

   Zunächst werden alle Modelle mit einem einfachen Zero-Shot-Prompt in unterschiedlichen Konfigurationen auf dem Developmentdatensatz ausgeführt. Diese Baseline dient der Identifikation offensichtlicher Fehlerquellen und liefert einen ersten Überblick über die Leistungsfähigkeit der Strategien.

   In diesem frühen Stadium wird auch der Einfluss der verwendeten Textextraktionsbibliothek untersucht, da die Qualität der aus den PDFs gewonnenen Textbasis unmittelbaren Einfluss auf die Modellleistung hat. Mehrere Bibliotheken werden auf einer Teilstichprobe verglichen (vgl. Abschnitt 7.2.3). Auf Basis dieser Ergebnisse wird die für die finale Evaluation verwendete Bibliothek festgelegt.

2. **Iterative Optimierung**

   Auf Basis der initialen Ergebnisse werden Prompts und Konfigurationen gezielt optimiert. Der Developmentdatensatz ist bewusst komplex gestaltet (vgl. Abschnitt 5.4), um Schwächen früh sichtbar zu machen. Während dieser Phase werden Zero- und Few-Shot-Ansätze mit zusätzlichem CoT-Prompting untersucht.

3. **Finale Evaluation**

   Nach Abschluss der Optimierung werden die final definierten Strategien unverändert auf dem unabhängigen Evaluationsdatensatz ausgeführt. Die strikte Trennung der Datensätze wahrt den Blind-Test-Charakter und folgt damit dem in Abschnitt 5.4 beschriebenen Evaluationsprinzip.

Alle Ergebnisse werden automatisiert in der Plattform persistiert, feldspezifisch ausgewertet und anhand der in Abschnitt 5.3 beschriebenen Metriken verglichen. Nach Beginn der finalen Evaluation werden keine Änderungen mehr an den Strategien vorgenommen, um eine unverfälschte Bewertung sicherzustellen.

6 Evaluationsplattform

Die Evaluation verschiedener LLM-basierter Extraktionsansätze erfordert eine technische Infrastruktur, die große Datenmengen effizient verarbeitet und einen reproduzierbaren Versuchsablauf sicherstellt. Aus diesem Grund wurde im Rahmen dieser Arbeit eine dedizierte Evaluationsplattform entwickelt, die sämtliche Experimente ausführt, verwaltet und auswertet. Sie ist gezielt auf die Sicherstellung von Reproduzierbarkeit, Vergleichbarkeit und Nachvollziehbarkeit der Verarbeitungsschritte ausgerichtet und bildet damit ein Forschungsartefakt im Sinne der gestaltungsorientierten Wirtschaftsinformatik [66].

Die Plattform ist als Webanwendung auf Basis des Ruby-on-Rails-Frameworks [80] implementiert und verwendet PostgreSQL als relationale Datenbank zur strukturierten und konsistenten Speicherung sämtlicher Evaluationsdaten. Diese Architektur unterstützt die parallele Verarbeitung großer Dokumentenmengen, gewährleistet eine konsistente Datenspeicherung und ermöglicht eine lückenlose Nachvollziehbarkeit aller Verarbeitungsschritte. Die Organisation der Funktionalität erfolgt über zentrale Domänen-Entitäten, die verschiedene Aspekte des Evaluationsprozesses abbilden und ein vollständiges System zur Bewertung von Extraktionsverfahren bereitstellen.

Alternative Ansätze, die auf Tabellenkalkulationen oder CSV-Dateien beruhen, stoßen bei der Verarbeitung von mehreren zehntausend Dokumenten schnell an ihre Grenzen, insbesondere in Bezug auf Performance, Automatisierung und strukturierten Vergleich [81]. Auch Jupyter-Notebooks bieten hierfür keine geeignete Basis, da ihr Schwerpunkt stärker auf explorativer, interaktiver Arbeit liegt und sie weniger auf eine standardisierte und wiederholbare Versuchs-durchführung ausgerichtet sind [82].

Besondere Merkmale der Evaluationsplattform sind ein flexibles Strategy-Konzept zur Anwendung unterschiedlicher Extraktionsansätze sowie die Integration mehrerer Textextraktionsbibliotheken mit Fallback-Mechanismen. Diese Funktionen sind speziell auf die Anforderungen einer wissenschaftlichen Evaluation zugeschnitten und unterstützen die in Kapitel 5 formulierten methodischen Ziele.

6.1 Beschreibung der Evaluationsplattform

Die Evaluationsplattform ist das technische Kernstück der Untersuchung und dient der Bewertung verschiedener Extraktionsverfahren. Sie ermöglicht, unterschiedliche LLM-Strategien auf denselben Datensätzen auszuführen, die Ergebnisse automatisch untereinander zu vergleichen und relevante Performance-Metriken zu berechnen. Das System folgt einem dokumentenzentrierten Ansatz, bei dem jedes Rechnungsdokument durch verschiedene Extraktionsstrategien verarbeitet und die Resultate mit den vorhandenen Ground-Truth-Daten abgeglichen werden. Die einzelnen Funktionsbereiche der Plattform erfüllen jeweils eine spezifische methodische Aufgabe. Die Dokumentenübersicht schafft Transparenz über alle eingespielten Belege und stellt damit die Nachvollziehbarkeit sicher. Der Strategiebereich ermöglicht das Anlegen neuer Strategien beziehungsweise Prompts, zeigt erste Kennzahlen und eröffnet Optionen für detailliertere Auswertungen, wodurch Vergleichbarkeit und systematische Analyse unterstützt werden. Der Sample-Set-Bereich liefert Kennzahlen zu den jeweiligen Datensätzen und trägt damit zur methodischen Planung und Kontrolle der Stichprobenbasis bei.

Darüber hinaus weist die Plattform zentrale Eigenschaften einer „Experimentation Workbench" auf, wie sie in der Softwaretechnik beschrieben wird [67]. Sie adressiert die dort formulierten Kernanforderungen: (R1) Unterstützung beim Setup von Experimenten durch die klare Definition von Strategien und Sample-Sets, (R2) Analyse von Ergebnissen über automatisiert erzeugte Leistungskennzahlen, (R3) einfache Variation von Experimenten durch erneute Ausführungen mit identischen oder modifizierten Parametern, (R4) umfassende Dokumentation aller Artefakte und Ergebnisse in einer konsistenten Datenbank, (R5) Wiederverwendung von Experimentkonfigurationen durch persistente Speicherung und erneute Ausführung sowie (R6) Erweiterbarkeit durch Integration neuer Strategien oder zusätzlicher Extraktionsverfahren. Damit erfüllt die Plattform sowohl den Anspruch, wissenschaftliche Strenge und praktische Relevanz zu verbinden, als auch die methodischen Anforderungen an eine „Experimentation Workbench".

6.1.1 Dokumentenübersicht

Die Dokumentenübersicht dient zur Verwaltung und Analyse aller PDF-Rechnungsdokumente innerhalb der Evaluationsplattform. Sie gewährleistet, dass alle Experimente auf einer identischen, kontrollierten Dokumentbasis stattfinden, und besteht aus zwei Hauptansichten:

Index-Ansicht:

Die Index-Ansicht listet alle verfügbaren Dokumente auf und bietet verschiedene Filteroptionen, um gezielt Dokumente zu finden. Gefiltert werden kann nach Sample Set, Kreditor, angewendeter Strategie, vordefinierten Fehlertypen sowie nach markierten Favoriten. Die Übersicht zeigt für jedes Dokument grundlegende Informationen wie Kreditor, Anzahl vorhandener Extraktionen und ob sie als favorisiert markiert wurde, was im Zuge der Evaluation für markante Fehler oder komplexe Rechnungen verwendet wird. Von hier aus kann direkt über „View" in die Detailansicht gewechselt werden.

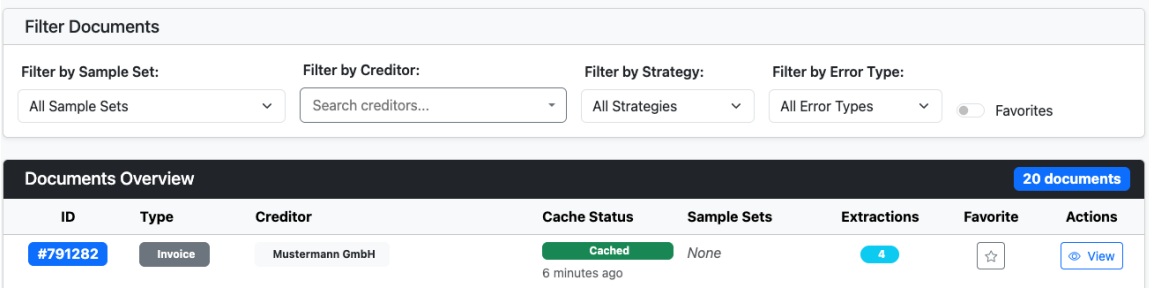


Abbildung 6.1 Document Index – Filtermöglichkeiten und Dokumentenübersicht

Show-Ansicht:

In der Show-Ansicht werden die Inhalte und Ergebnisse eines einzelnen Dokuments dargestellt. Sie ist in drei wesentliche Bereiche unterteilt:

- **Vergleichsansicht**: Zeigt die von verschiedenen Strategien extrahierten Felder (Rechnungsnummer, Rechnungsbetrag, Rechnungsdatum) im direkten Vergleich zur Ground Truth, GINI-Ergebnissen und LLM-Extraktionen.
- **Dokumentviewer**: Stellt das Original-PDF dar, um die extrahierten Informationen direkt zu überprüfen.
- **Extraktionsdetails**: Extraktionen werden als Text bzw. als JSON gespeichert. Alle Extraktionen, also LLM-Extraktionen, Ground-Truth-Daten, Textextraktionen und Gini-Extraktionen, können hier eingesehen werden.

Über Buttons wie „Re-Extract Text", „Re-Process All" und „Process" können Textextraktionen oder LLM-Läufe erneut gestartet werden. So lassen sich Strategien optimieren oder neue Ergebnisse erzeugen. Änderungen an den Ground-Truth-Daten werden zudem versioniert gespeichert, sodass fehlerhafte Annotationen im Nachhinein ausgewertet werden können.

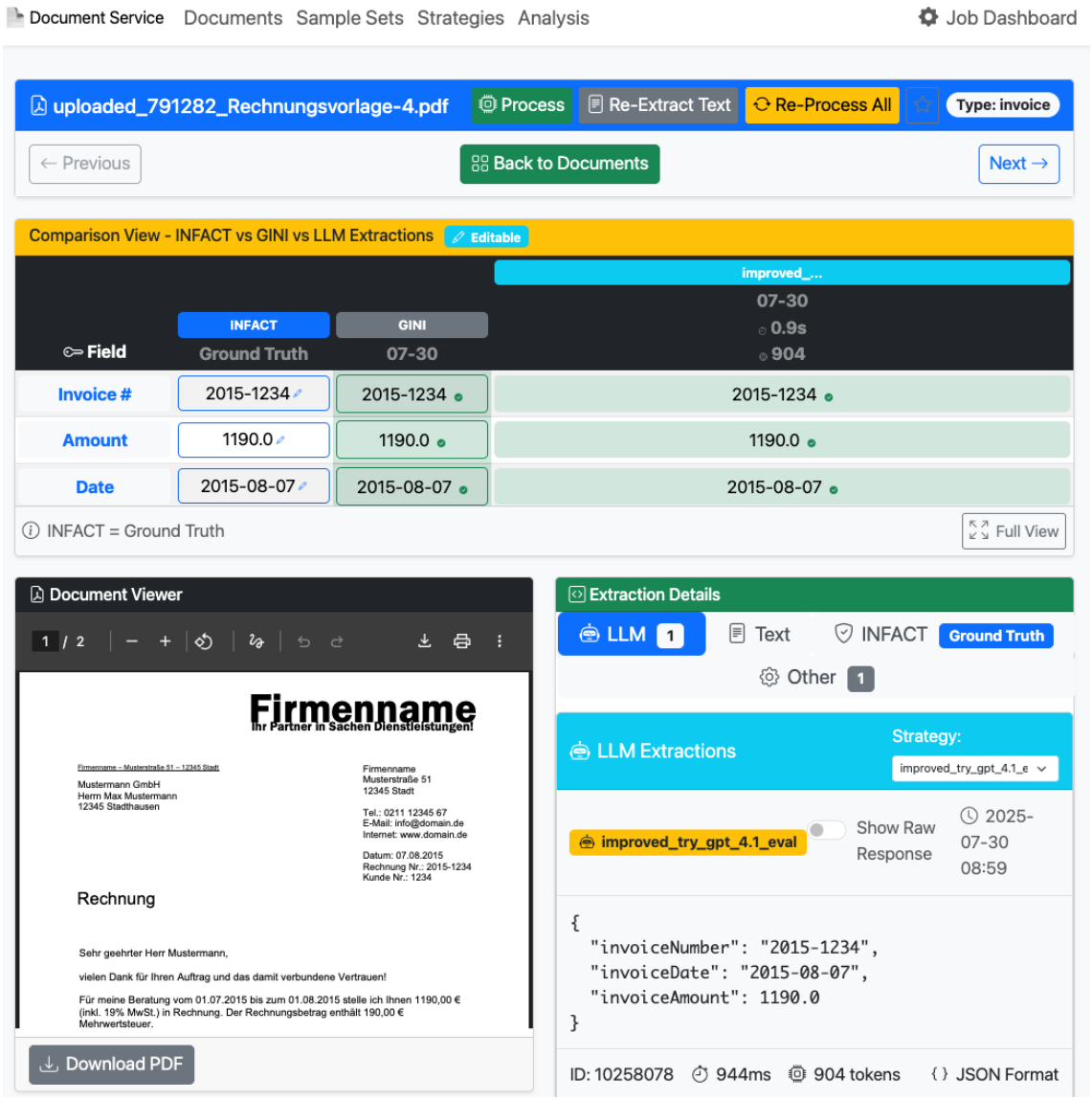


Abbildung 6.2 Document Show – Vergleichsansicht, PDF-Dokumentviewer und Extraktionsdetails einer einzelnen Rechnung

6.1.2 Strategieübersicht

Der Strategiebereich dient zur Verwaltung und Auswertung aller definierten LLM-basierten Extraktionsstrategien. Es stellt die Vergleichbarkeit sicher, indem jede Strategie mit dokumentierten Parametern auf identischen Datensätzen ausgeführt wird, und besteht aus den folgenden Bereichen:

Index-Ansicht:

Die Index-Ansicht listet alle vorhandenen Strategien auf und bietet Filtermöglichkeiten nach LLM-Provider, Modell und Extraktionsmethode. Zusätzlich können versteckte Strategien ein- oder ausgeblendet werden. Die Übersicht zeigt für jede Strategie grundlegende Informationen wie ID, Name, eingesetztes Modell, verwendete Methode, Anzahl durchgeführter Extraktionen und den Status (aktiv oder verborgen). Hier können neue Strategien erstellt, bestehende bearbeitet oder Ergebnisse eingesehen werden. Über die Schaltfläche „Compare Performance" kann ein direkter Leistungsvergleich mehrerer Strategien gestartet werden.

Compare Performance:

Die Vergleichsansicht ermöglicht es, mehrere Strategien gleichzeitig zu analysieren. Sie bietet eine Auswahl der zu vergleichenden Strategien und Sample Sets sowie verschiedene Visualisierungen:

- **Overall Performance Chart**: Darstellung der Gesamtgenauigkeit pro Strategie.
- **Field Performance Breakdown**: Feldspezifische Genauigkeit für Rechnungsnummer, Datum und Betrag.
- **Overall Performance Ranking**: Tabellarische Auswertung mit Gesamtwerten, $Overall\ Accuracy$, $Dokument - Accuracy$, $Feld - Accuracy$, Tokenverbrauch und Anzahl der Vergleiche.
- **McNemar-Test**: statistischer Vergleich zweier Strategien auf Dokument- und Feldebene inklusive Kontingenztabelle, p-Werten und Odds-Ratios.

Show-Ansicht:

Ebenfalls über die Indexansicht ist die Detailansicht einer einzelnen Strategie zu erreichen. Sie ist in mehrere Bereiche unterteilt:

- **Ausführungsübersicht**: Zeigt den aktuellen Verarbeitungsstatus, die Anzahl der verarbeiteten Dokumente, Erfolgsrate, Gesamtgenauigkeit sowie den durchschnittlichen Tokenverbrauch.
- **Feldgenauigkeit**: Stellt die Genauigkeit pro Feld dar, um Stärken und Schwächen der Strategie schnell erkennen zu können.
- **Strategy Details**: Listet die verwendeten Parameter wie Modell, Temperature, Top-P, Max Tokens und Vision-Einstellungen auf.
- **Sample-Set-Statistiken**: Zeigt, wie viele Dokumente je Sample Set verarbeitet wurden und mit welcher Genauigkeit.

Über die Funktion „Batch Process" können alle offenen oder alle Dokumente eines Sample Sets gestartet werden. Mit „Cancel Jobs" lassen sich laufende Jobs abbrechen, und über „Advanced Statistics" werden zusätzliche Auswertungen auf Kreditor-Ebene sowie die Ergebnisse des McNemar-Tests zwischen Strategie- und Gini-Ergebnissen anzeigen.

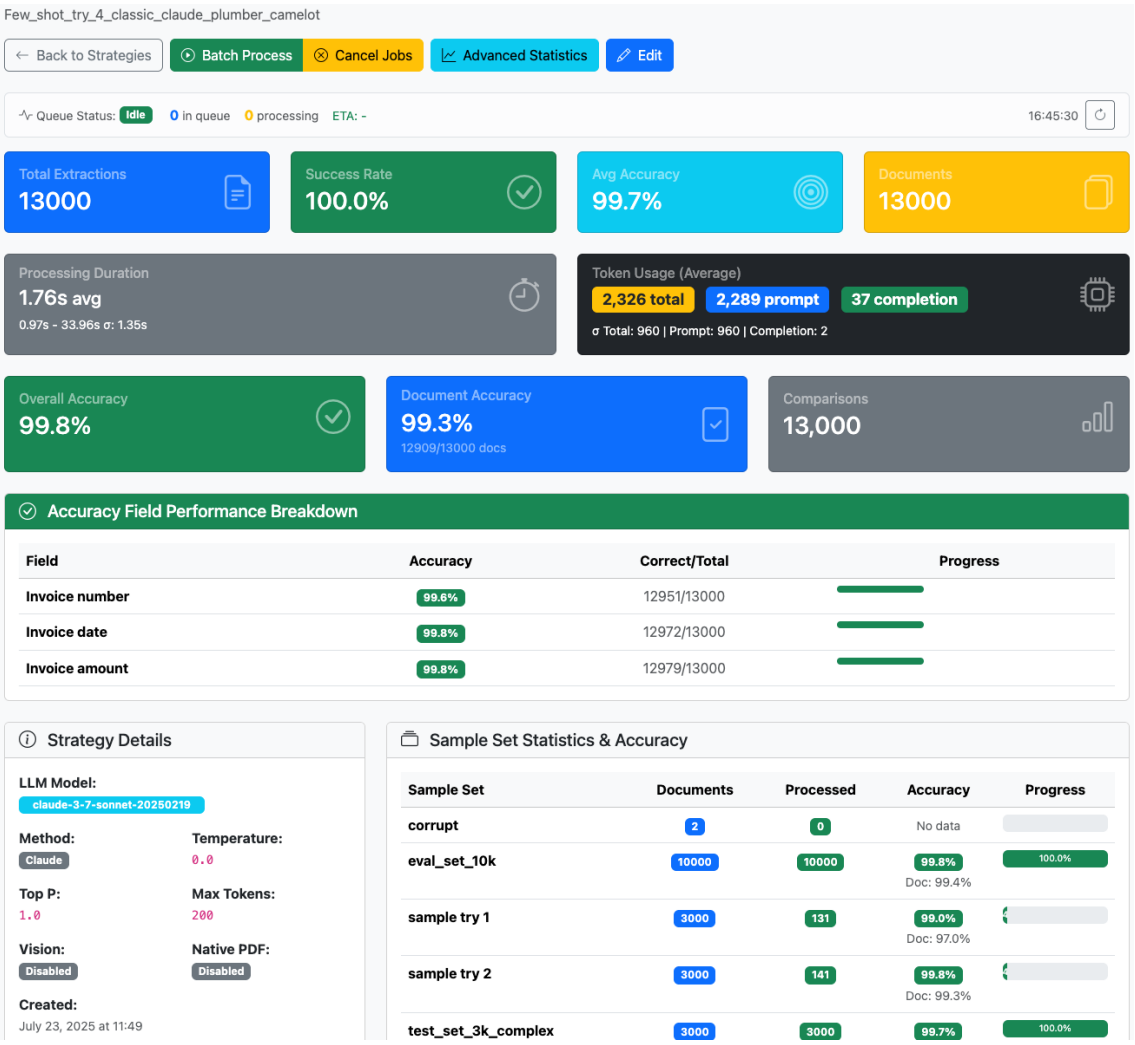


Abbildung 6.3 Strategy Show – Übersicht einer Strategie mit Verarbeitungsstatus, Feldgenauigkeit und Parametern

6.1.3 Sample-Set-Bereich

Der Sample-Set-Bereich ermöglicht eine Übersicht über Dokumentengruppen für gezielte Testreihen und gliedert sich in folgende Teile:

Index-Ansicht:

Die Index-Ansicht listet alle vorhandenen Sample-Sets auf. Für jedes Set werden Name und Anzahl der enthaltenen Dokumente angezeigt. Über „View Details" können Detailinformationen abgerufen werden. In der Systemübersicht wird zusätzlich die Gesamtanzahl der vorhandenen Datensätze und der darin befindlichen Dokumente angezeigt.

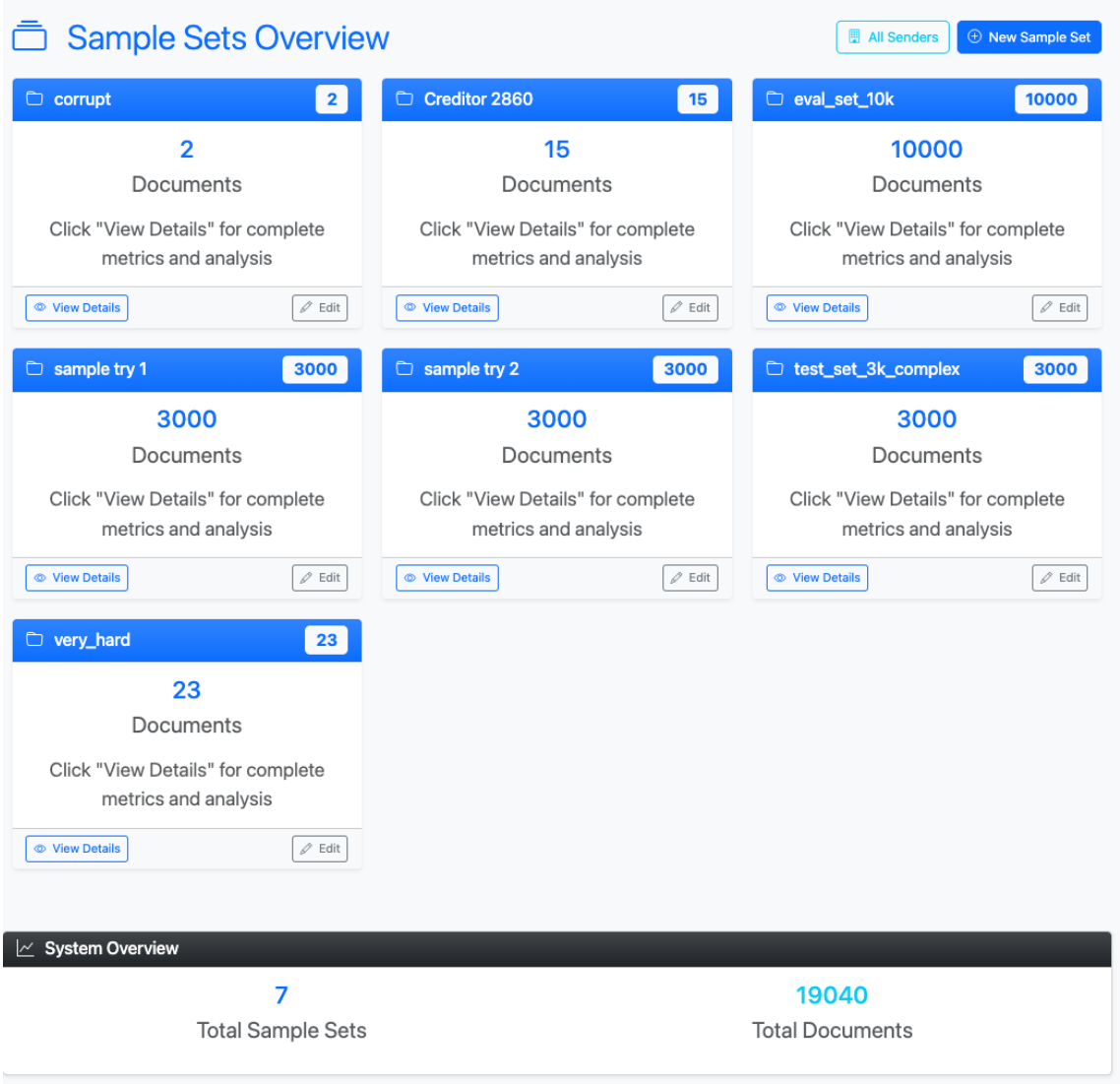


Abbildung 6.4 Sample Set Index – Übersicht aller verfügbaren Sets mit Dokumentenzahlen

Show-Ansicht:

In der Detailansicht eines Sample-Sets werden zusätzliche Informationen angezeigt:

- **Sample Set Information**: Gesamtzahl der enthaltenen Dokumente, Anzahl der LLM-Extraktionen, Infact-Extraktionen und die Anzahl der unterschiedlichen Kreditoren.
- **Top Sender**: Auflistung der Kreditoren mit Angabe, wie viele Dokumente je Kreditor enthalten sind.

Über die Funktion „Bulk Re-Extract Text" können für alle Dokumente im Set erneut die Texte extrahiert werden, wobei die gewünschte Textextraktion ausgewählt werden kann. Ergänzend ermöglicht die Funktion „Ground Truth Impact" die Analyse der Auswirkungen von Änderungen am Ground Truth. Über einen Button in der Detailansicht wird ein Modal geöffnet, das die GINI-Extraktionsergebnisse sowohl mit den ursprünglichen als auch mit den angepassten Werten vergleicht und damit sichtbar macht, wie stark Korrekturen einzelner Felder die Feld- und $Dokument\ Accuracy$ beeinflussen. Neben aggregierten Kennzahlen werden auch feldspezifische Auswirkungen sowie betroffene Dokumente mit ihren jeweiligen Änderungen angezeigt.

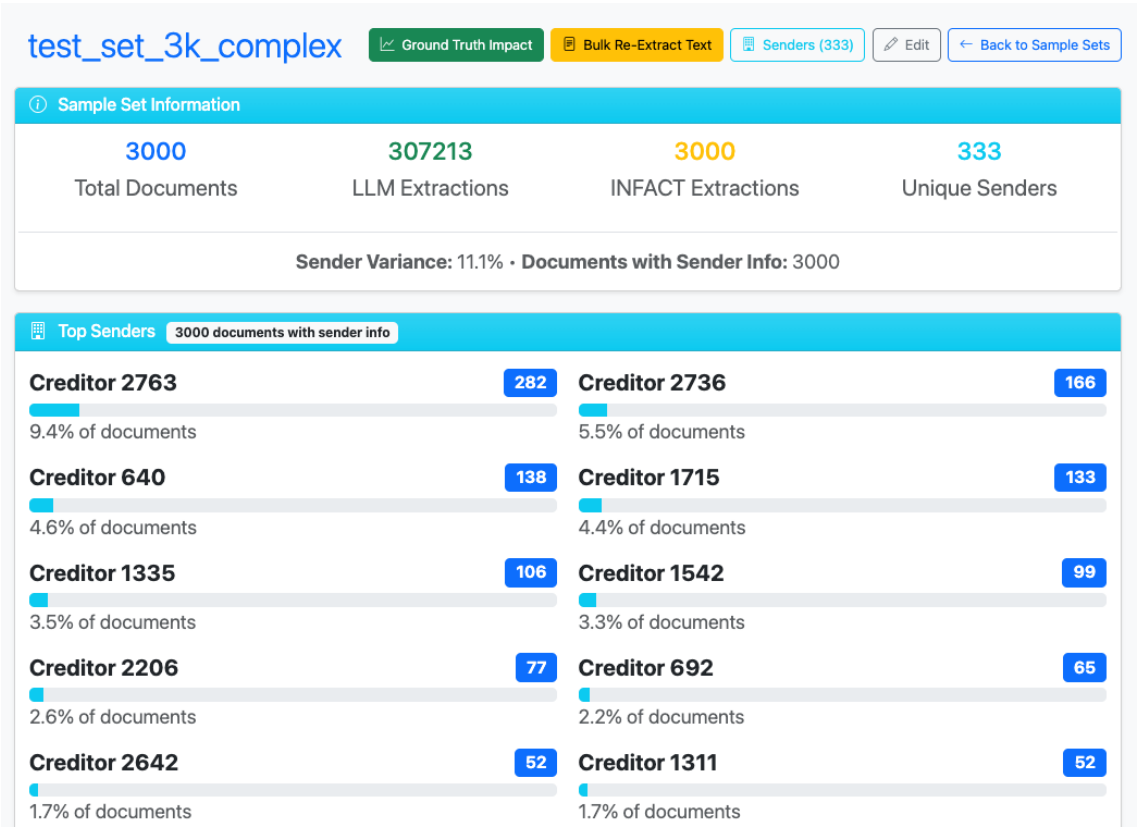


Abbildung 6.5 Sample Set Show – Detailansicht mit Dokumentstatistik und Kreditorenübersicht

6.2 Datenbank-Architektur

Die Evaluationsplattform greift für die Durchführung der Experimente auf die in Kapitel 5.2 beschriebene Datengrundlage zurück und überführt diese in eine für die Experimente geeignete technische Struktur. Dazu werden die für die Evaluation benötigten Rechnungsdaten aus der relationalen Datenbank des Factoring-Systems Infact (Abschnitt 5.2) in ein einheitliches JSON-Format exportiert und anschließend in die Evaluationsplattform importiert. Dieses JSON bildet sämtliche für aifinyo relevanten Rechnungsinformationen ab, auch über die drei untersuchten Kernfelder Rechnungsnummer, Rechnungsbetrag und Rechnungsdatum hinaus, und stellt damit eine konsistente Datengrundlage für alle weiteren Auswertungen bereit. Sowohl die ursprünglichen Gini-OCR-Extraktionen als auch die Infact-Referenzdaten werden in dieser Form übernommen und persistiert, was eine einheitliche Verarbeitung und einen direkten Vergleich der Ergebnisse ermöglicht, wie in Abbildung 6.6 dargestellt.


```json
{
  "invoiceNumber": "RE-20150505",
  "invoiceDate": "2025-09-19",
  "invoiceAmount": 354.80,
  "invoiceAmountUst": [
    {
      "rate": "19%",
      "amount": 101.134
    }
  ],
  "targetDays": null,
  "sender": {
    "name": "Muster GmbH",
    "street": "Musterweg",
    "streetNumber": "43",
    "zip": "12345",
    "city": "Musterdstadt"
  },
  "recipient": {
    "name": "Max Mustermann",
    "street": "Musterstraße",
    "streetNumber": "1",
    "zip": "12345",
    "city": "Musterstadt"
  }
}
```


Abbildung 6.6 Beispielhafte JSON-Struktur einer Rechnung (vollständiges JSON siehe Anhang 1.1)

Für die Persistenz der Daten wird eine PostgreSQL-Datenbank eingesetzt, die sich insbesondere durch ihre native JSON-Unterstützung und hohe Stabilität bewährt hat [83], [84]. Dies geschieht in sechs Tabellen, die den gesamten Evaluationsprozess abbilden (Abbildung 6.7). Systeminterne Tabellen werden im Folgenden nicht berücksichtigt. Die wesentlichen Strukturen sind:

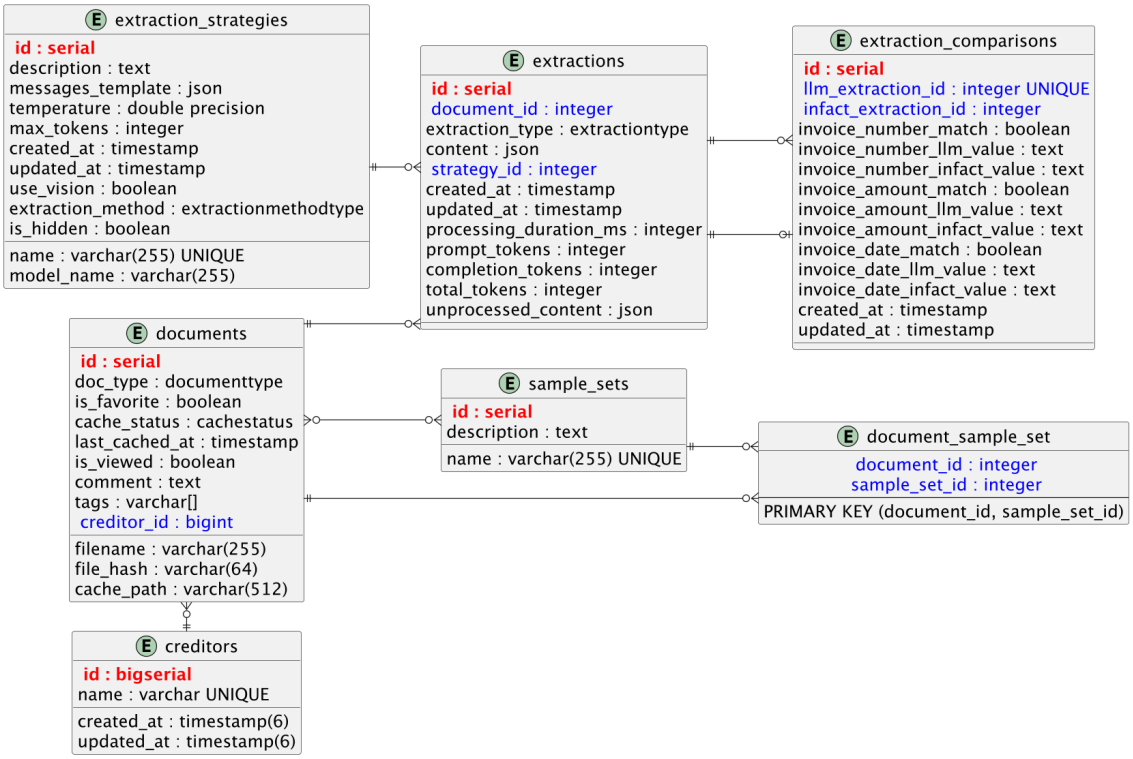


Abbildung 6.7 Datenbankstruktur der Evaluationsplattform

Documents: Die Documents-Tabelle bildet das Herzstück der Datenbank und beinhaltet alle Rechnungsdokumente, einschließlich des Dokumentennamens, einen Verweis auf den Kreditoren sowie eines kryptographischen Hashes des PDFs zur eindeutigen Identifikation. Darüber hinaus werden Metadaten zur Cache- und Statusverwaltung persistiert, um eine konsistente Verarbeitung und Nachverfolgbarkeit sicherzustellen.

Creditors: Diese Tabelle bildet die Kreditoren (Rechnungsersteller) ab und wird vor allem für Filterungen und Auswertungen genutzt.

ExtractionStrategy: Die ExtractionStrategies-Tabelle definiert die verschiedenen LLM-Ansätze mitsamt ihren spezifischen Konfigurationen, wie Modellvarianten, Prompt-Templates, $temperature$, top_p und $max - token$. Jede Strategie repräsentiert eine versionierte, eindeutig referenzierbare Extraktionskonfiguration.

Extraction: Die Extractions-Tabelle speichert alle Verarbeitungsergebnisse mit präzisen Zeitstempeln, Token-Verbrauch und Laufzeiten. Die Extraktionsergebnisse werden sowohl in verarbeiteter als auch in unverarbeiteter Form im JSON-Format gespeichert und ermöglichen eine Nachverfolgung des gesamten Evaluationsprozesses. Jede Extraction ist einer ExtractionStrategy und einem Document zugeordnet. Weiterhin werden auch die historischen Rechnungs-OCR-Extraktionen sowie die Werte aus der Factoring-Software Infact in der Tabelle abgelegt.

ExtractionComparison: Die ExtractionComparisons-Tabelle führt die automatisierte Bewertung durch und vergleicht Extraction-Ergebnisse mit den Infact-Referenzdaten. Diese Tabelle speichert sowohl die Vergleichsergebnisse als auch die berechneten Performance-Metriken und wird durch eine Trigger-Function automatisch befüllt.

SampleSet: Die SampleSets-Tabelle ermöglicht die flexible Gruppierung von Dokumenten für spezifische Experimente. Hierüber werden Development- und Evaluationsdatensätze abgebildet.

DocumentSampleSet: Die DocumentSampleSets-Tabelle fungiert als Verbindungstabelle zwischen Document und SampleSet und realisiert die Many-to-Many-Beziehung für die flexible Dokumentengruppierung. Diese Struktur stellt über diese Entitäten sicher, dass jedes Dokument durch verschiedene Strategien verarbeitet und gleichzeitig in beliebigen Sample-Sets organisiert werden kann.

6.3 Funktionelle Besonderheiten

Die Evaluationsplattform enthält mehrere technische Funktionen, die den Ablauf der Experimente unterstützen und für reproduzierbare Ergebnisse sorgen. Ein wesentlicher Bestandteil ist die Nutzung von PostgreSQL-Trigger-Functions [85]. Dadurch werden Berechnungen und Vergleiche direkt in der Datenbank ausgeführt, sodass Metriken automatisch aktuell gehalten und große Datenmengen effizient verarbeitet werden können. Ein weiterer Schwerpunkt liegt im flexiblen Umgang mit Extraktionsstrategien. Diese können mit verschiedenen Modellen, Parametern und Prompts konfiguriert und systematisch auf Datensätze angewendet werden. Dadurch lassen sich unterschiedliche Ansätze einfach testen und vergleichen. Zusätzlich stehen mehrere Text-Extraktionsbibliotheken und LLM-Anbindungen zur Verfügung, deren Ergebnisse innerhalb der Plattform miteinander verglichen werden können, um Unterschiede in Leistungsfähigkeit und Robustheit zu untersuchen.

6.3.1 Trigger für performante Auswertungen

Um die Ergebnisse verschiedener Extraktionsmethoden zuverlässig und ohne manuelle Nacharbeit vergleichen zu können, nutzt die Evaluationsplattform PostgreSQL-Trigger-Functions [85]. Bei jeder neuen oder aktualisierten Extraktion wird automatisch eine Vergleichsfunktion aufgerufen, die die relevanten Felder (Rechnungsnummer, Betrag, Datum) zwischen LLM-Ergebnissen, Infact-Referenzdaten und gegebenenfalls historischen Rechnungs-OCR-Ergebnissen überprüft.

Die Trigger sorgen dafür, dass Vergleichsdaten sofort in der Datenbank aktualisiert werden, ohne dass separate Verarbeitungsschritte notwendig sind. Dadurch stehen Metriken nahezu in Echtzeit zur Verfügung, selbst bei großen Datenmengen. Gleichzeitig wird vermieden, dass alle Dokumente global neu berechnet werden müssen, denn nur die betroffenen Datensätze werden geprüft und aktualisiert.

Die Automatisierung erfolgt über eine zentrale Trigger-Funktion, die beim Einfügen oder Aktualisieren einer Extraktion ausgelöst wird. Sie ruft für die betroffenen Dokumente Vergleichsoperationen auf und schreibt die Resultate in eine Vergleichstabelle, sodass Abweichungen zwischen LLM-, Infact- und gegebenenfalls GINI-Ergebnissen unmittelbar nachvollziehbar sind (Abbildung 6.8).

```
Trigger: on INSERT or UPDATE of an extraction
    if extraction type is relevant (LLM, GINI, INFACT)
        call recalculate_extraction_comparison(extraction_id)

Function recalculate_document_extraction_comparisons(document_id):
    for each extraction of type LLM or GINI in this document
        recalculate_extraction_comparison(extraction_id)
    for each extraction of type INFACT in this document
        recalculate_extraction_comparison(extraction_id)

Function recalculate_extraction_comparison(extraction_id):
    get extraction_row by id

    if extraction_row is LLM or GINI:
        find latest INFACT extraction for same document
        if none exists → return
        extract invoiceNumber, invoiceAmount, invoiceDate from both
        if comparison already exists:
            update comparison record with match results and values
        else:
            insert new comparison record with values

    else if extraction_row is INFACT:
        for each LLM or GINI extraction of same document
            recalculate_extraction_comparison(extraction_id)
```

Abbildung 6.8 Pseudocode der Trigger-Logik zur Neuberechnung von Extraktionsvergleichen (vollständiges JSON siehe Anhang 2.1)

Zu beachten ist jedoch, dass diese Funktionalität Logik, die normalerweise im Anwendungscode zu erwarten wäre, in die Datenbank verlagert. Dadurch entsteht zwar ein gewisser Performance-Overhead auf Datenbankebene, wie in der Literatur als Nachteil von Triggern beschrieben wird [86], gleichzeitig entfällt jedoch die Notwendigkeit zusätzlicher Verarbeitungsschritte außerhalb der Datenbank. Im Kontext der Evaluationsplattform überwiegt damit der Vorteil einer höheren Automatisierung und unmittelbaren Reproduzierbarkeit.

6.3.2 Strategy-Konzept und Sample-Execution

Die Evaluationsplattform stellt ein flexibles System bereit, um verschiedene Extraktionsstrategien zu definieren und auf Dokumentmengen anzuwenden. Eine Strategie bündelt alle relevanten Parameter für einen Extraktionslauf zusammen, wie das verwendete Modell, die LLM-spezifischen Parameter ($temperature$, $top-p$, $max-tokens$) und das jeweilige Prompt-Template. Abbildung 6.9 veranschaulicht beispielhaft die Konfiguration einer solchen Strategie.

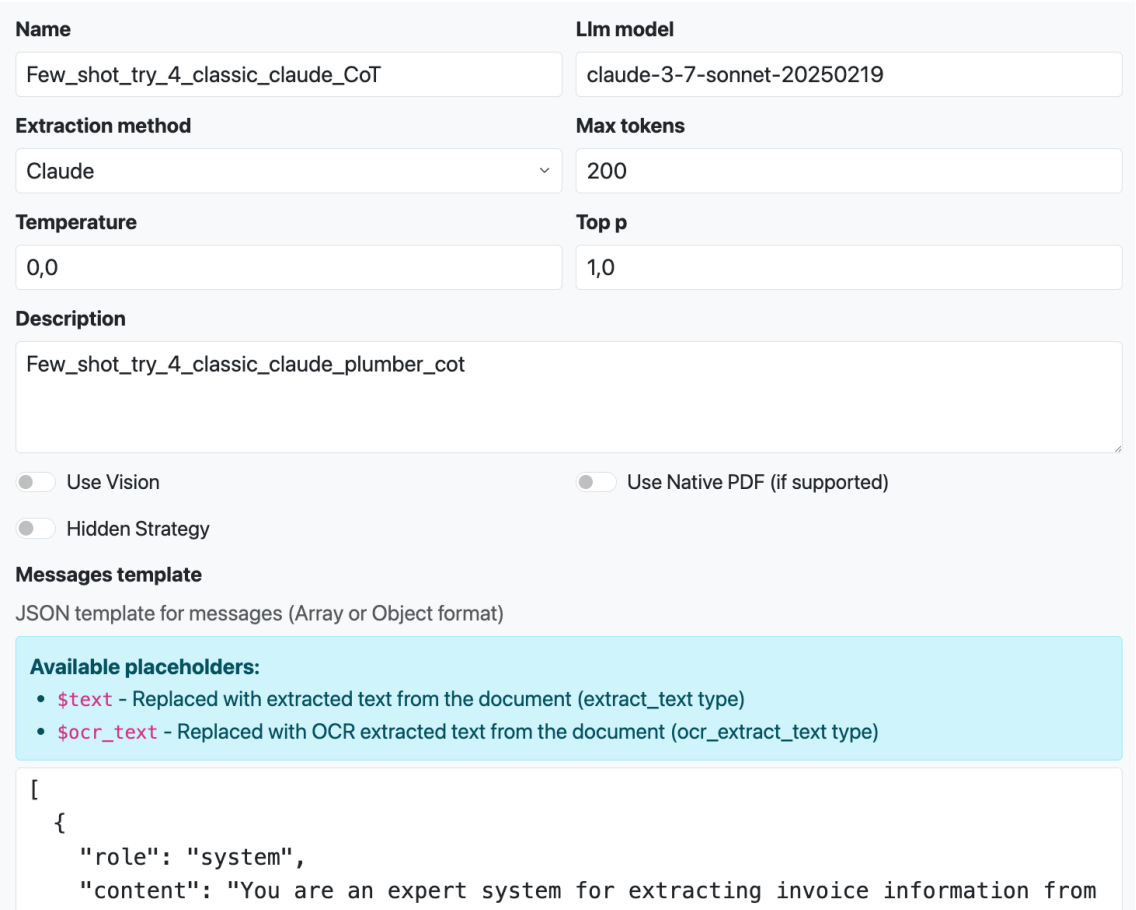


Abbildung 6.9 Beispiel-Dialog zur Konfiguration einer Extraktionsstrategie (Strategie-Name, Modell, Parameter und Prompt-Template)

Prompts können dabei Platzhalter enthalten, die beim Ausführen der Strategie automatisch mit den Inhalten des jeweiligen Dokuments befüllt werden. Dazu stehen vordefinierte Variablen wie $\$text$ (Textextraktion) oder $\$ocr\_text$ (OCR-Textextraktion) zur Verfügung. So kann dieselbe Konfiguration für beliebig viele Dokumente genutzt und systematisch getestet werden.

Die Ausführung der Strategien erfolgt über ein Queue-System. Jedes Dokument des gewählten Sample Sets wird dabei an die LLM-Schnittstelle übergeben und das Ergebnis in der Datenbank gespeichert. Bei Fehlern kommen automatische Wiederholungsversuche mit Backoff-Strategie (schrittweise verlängerte Wartezeit zwischen erneuten Anfragen) zum Einsatz, um eine möglichst vollständige Verarbeitung zu erreichen.

6.3.3 Text Extraction Libraries

Die Textextraktion aus PDF-Dokumenten ist ein entscheidender Schritt vor der eigentlichen LLM-Verarbeitung. Fehler oder unvollständige Texte in dieser Phase wirken sich direkt auf die Qualität der Ergebnisse aus. Daher bietet die Evaluationsplattform mehrere Extraktionsbibliotheken, die verwendet werden können.

Die Auswahl erfolgt manuell über die Benutzeroberfläche (siehe Abbildung 6.10). Aktuell stehen folgende Optionen zur Verfügung:

Bibliothek	**Version**	**Konfiguration & Settings**
Pdfplumber [87]	0.7.0	**Extraktionsmethode**: page.extract_text() **Layout-Parameter**: Keine **Fallback-Schwelle**: 20 Zeichen pro Seite **Fallback-Mechanismus**: PyMuPDF (~1%)
pdfminer.six [22]	20221105	**Extraktionsmethode**: extract_text() **Layout-Parameter**: LAParams() **Fallback-Schwelle**: 20 Zeichen pro Seite **Fallback-Mechanismus**: PyMuPDF (~1%)
PyMuPDF [88]	1.23.0	**Extraktionsmethode**: page.get_text() **Layout-Parameter**: Keine Blöcke/Strukturierung **Fallback-Schwelle**: Keine (konnte jedes Dokument auslesen)
pypdfium2 [89]	4.18.0	**Extraktionsmethode**: get_textpage() + get_text_range() **Layout-Parameter**: Standard-Lesereihenfolge **Fallback-Schwelle**: Keine (konnte jedes Dokument auslesen)

Tabelle 6.1a Textextraktions Bibliotheken

Bibliothek	Version	Konfiguration & Settings
pdf-reader (Ruby) [23]	2.14.1	**Extraktionsmethode**: page.text **Layout-Parameter**: Standard Layout-Optionen **Fallback-Schwelle**: Keine **Fallback-Mechanismus**: HexaPDF-Reparatur bei Fehlern
Tesseract OCR [25]	5.5.1	**Extraktionsmethode**: pytesseract.image_to_string() **Layout-Parameter**: --oem 3 --psm 6,3,4,1 (Fallback-Modi) **Fallback-Schwelle**: 5 Zeichen pro Seite **Fallback-Mechanismus**: Multiple PSM-Modi + Sprachen

Tabelle 6.1b Textextraktions Bibliotheken

Diese Bibliotheken können je nach Experiment und Dokumententyp gezielt eingesetzt werden. Durch den direkten Vergleich lassen sich Unterschiede in der Qualität und Robustheit der Textextraktion untersuchen, bevor die eigentliche LLM-Verarbeitung erfolgt. Als Standard-Fallback kommt PyMuPDF zum Einsatz, da es sich als besonders robust zeigt und alle vorliegenden Dokumente verarbeiten kann.

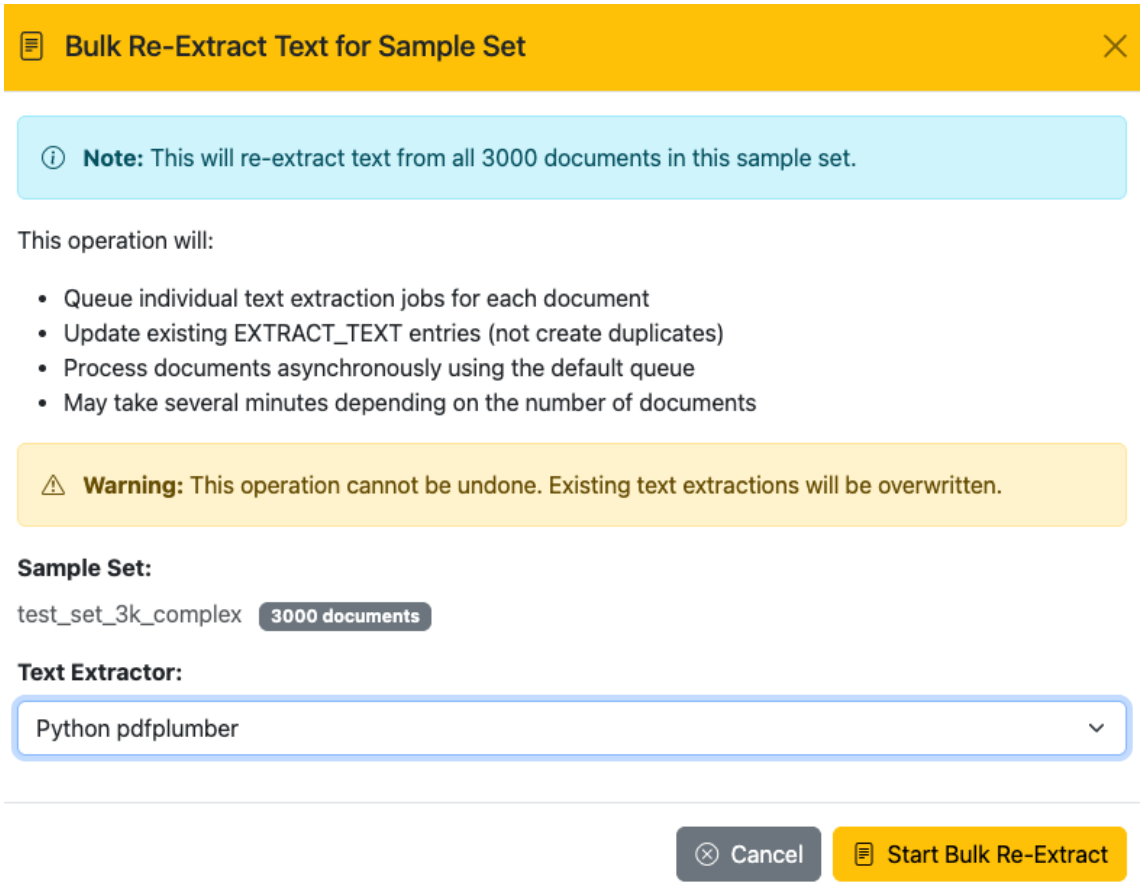


Abbildung 6.10 Auswahl des Text-Extraktors bei der erneuten Textextraktion für ein komplettes Sample-Set

6.4 Architektur

Die technische Architektur der Evaluationsplattform folgt einem modularen, serviceorientierten Aufbau und ist speziell für die Durchführung groß angelegter wissenschaftlicher Experimente mit Rechnungsdokumenten konzipiert. Sie kombiniert eine Rails-basierte Webanwendung mit asynchronen Hintergrundprozessen, die eine parallele und skalierbare Verarbeitung mehrerer Dokumente gleichzeitig ermöglichen.

Die Architektur ist klar in Schichten gegliedert und trennt Datenhaltung, Geschäftslogik und die Anbindung externer LLM-Dienste voneinander. Diese Struktur erleichtert nicht nur die Erweiterung um neue Extraktionsstrategien oder Analysefunktionen, sondern trägt auch zur Wartbarkeit und langfristigen Stabilität des Systems bei.

Durch die Service-Schicht und die einheitliche LLM-Anbindung werden die Verarbeitungsschritte transparent und reproduzierbar abgebildet. Damit schafft die Architektur die Grundlage für eine präzise und effiziente Evaluation unterschiedlicher LLM-basierter Extraktionsansätze.

6.4.1 Allgemeine Systemarchitektur

Die Evaluationsplattform ist als Ruby-on-Rails-Anwendung [80] mit modularem Schichtaufbau realisiert und unterstützt die Verarbeitung großer Mengen an Rechnungsdokumenten für Extraktions- und Vergleichsexperimente. Ziel der Architektur ist es, verschiedene LLM-Strategien systematisch anwenden und deren Ergebnisse effizient auswerten zu können.

Die Architektur folgt einem modularen Schichtaufbau, der Domain Model, Application Layer, Service Layer und Infrastruktur umfasst und in Abbildung 6.11 visualisiert ist. Das System basiert auf dem Model-View-Controller-Prinzip, ergänzt um eine Service-Schicht, die komplexe Logik wie die LLM-Anbindung oder asynchrone Hintergrundverarbeitung kapselt.

Für die parallele Verarbeitung großer Datenmengen werden Hintergrundjobs über Sidekiq [90] ausgeführt. Dadurch können mehrere tausend Dokumente verarbeitet werden, während die Webanwendung reaktionsfähig bleibt. Redis [91] übernimmt hierbei die Verwaltung der Queues und Caching-Aufgaben und bildet damit die Grundlage für eine skalierbare Hintergrundverarbeitung.

Die Architektur gliedert sich in vier Kernbereiche:

- **Domain Model**: Zentrale Objekte wie Dokumente, Extraktionen, Strategien und Vergleichsdaten.
- **Application Layer**: Rails-Controller und asynchrone Hintergrundprozesse zur Steuerung von Experimenten.
- **Service Layer**: Verwaltung der Anbindung externer LLM-Dienste über einen DI-Container mit dynamischer Auflösung spezifischer Services (z. B. OpenAI, Claude, LM Studio).
- **Infrastruktur**: PostgreSQL-Datenbank, Redis und lokaler Cache für effiziente Speicherung und Verarbeitung.

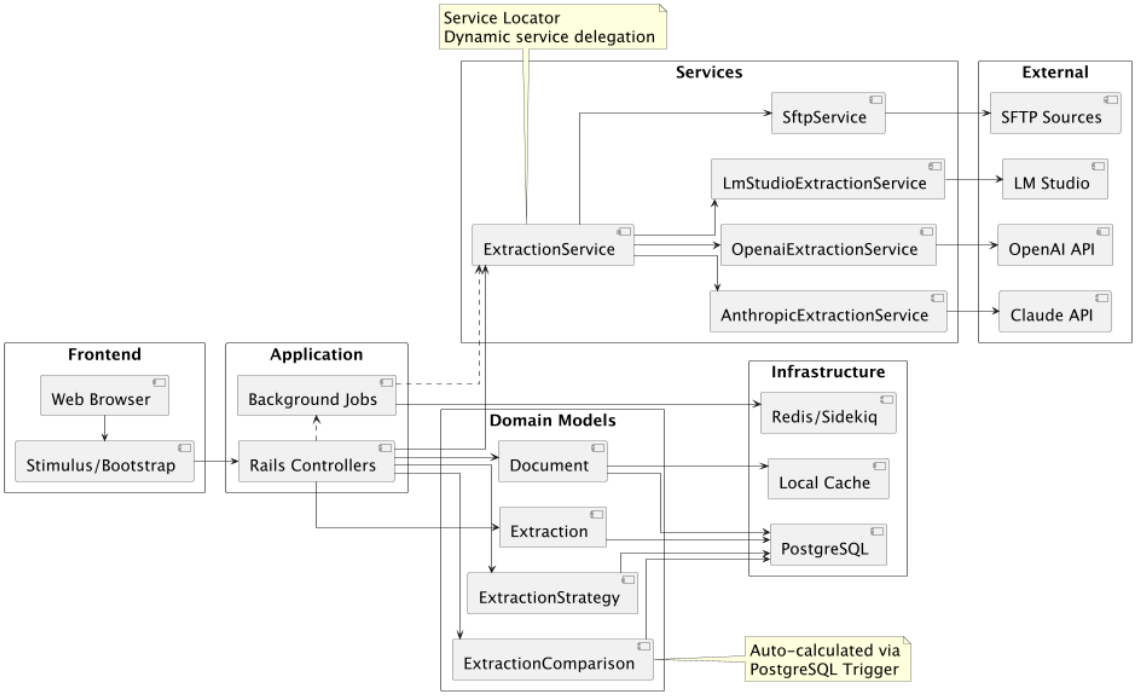


Abbildung 6.11 Systemarchitektur der Evaluationsplattform mit den zentralen Schichten und den angebundenen externen Diensten.

6.4.2 LLM Service Architektur

Die LLM-Services abstrahieren die Kommunikation mit verschiedenen Sprachmodellen und stellen eine einheitliche Schnittstelle für die Extraktion bereit. Wie in Abbildung 6.12 dargestellt, orchestriert der ExtractionCoordinator die Service-Auflösung über einen selbst implementierten, leichtgewichtigen DI-Container. Da Rails von Haus aus keinen klassischen DI-Container bietet, wird die dynamische Bereitstellung der Services hier über eine zentrale Mapping-Logik realisiert, was die modulare Integration unterschiedlicher LLM-Provider ermöglicht [92].

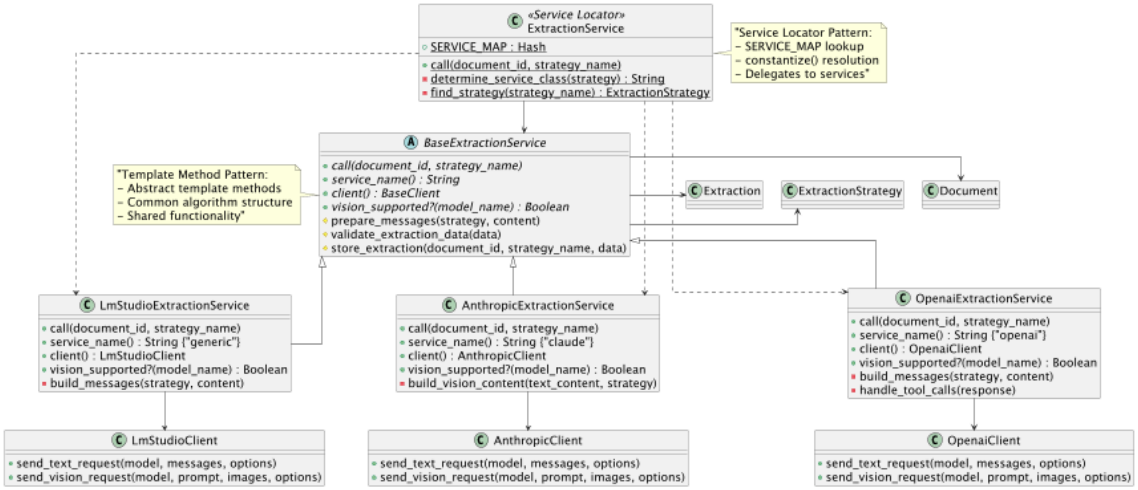


Abbildung 6.12 LLM-Services-Architektur

Die Services basieren auf einer abstrakten $BaseExtractionService$-Klasse, die gemeinsame Funktionalitäten wie JSON-Parsing, Fehlerbehandlung, und Datenvalidierung implementiert. Alle Services verwenden eine einheitliche $extract\_content$-Methode. Spezialisierte Services erben von dieser Basisklasse und kapseln eigene Client-Implementierungen für die spezifischen API-Anforderungen:

- **AnthropicExtractionService**: Claude-Integration
- **OpenaiExtractionService**: OpenAI GPT-Integration
- **LmStudioExtractionService**: Modelle über LMStudio

Die konkrete Implementierung dieser Service-Auflösung ist in Abbildung 6.13 dargestellt.

```
 1. class ExtractionCoordinator
 2.     def initialize(container: DiContainer.instance)
 3.       @container = container
 4.     end
 5.
 6.     def extract(document_id:, strategy_name:)
 7.       strategy = find_strategy(strategy_name)
 8.       service = @container.resolve(strategy.extraction_method)
 9.       service.call(document_id: document_id, strategy_name: strategy_name)
10.     end
11.   end
12.
13.   class DiContainer
14.     def resolve(key)
15.       service_class = @services[key.to_s]
16.       service_class.new
17.     end
18.   end
```

Abbildung 6.13 Implementierung der dynamischen Service-Delegation im ExtractionCoordinator

6.4.3 Response Parsing

Ein zentraler Bestandteil der Extraktionslogik ist die zuverlässige Verarbeitung der Modellantworten. Obwohl die Prompts ein eindeutiges JSON-Format vorschreiben, halten sich die LLMs in der Praxis nicht immer strikt daran.

Abbildung 6.14 bis 6.16 zeigen typische Abweichungen, die in den Antworten der Modelle auftreten können. Dazu zählen unnötige Array-Wrapper, mehrfach verschachtelte JSON-Objekte und fehlerhafte Klammerungen.

Antwort in einem überflüssigen Array

```
```json[ { "invoiceNumber": "4388", "invoiceDate": "2023-11-06", "invoiceAmount": 445.89
}]```
```

Abbildung 6.14 Erwartetes JSON in einem Array

Mehrere Objekte werden in einem Array zurückgegeben

```
```json[ { "invoiceNumber": "1085", "invoiceDate": "2022-11-10", "invoiceAmount": 18363.01
}, { "invoiceNumber": "00-5547", "invoiceDate": "2022-10-01", "invoiceAmount": 98.10 },...
```

Abbildung 6.15 Mehrere korrekte JSONs in einem Array

Fehlerhafte Klammerung mit überzähligen Zeichen am Ende, wodurch die JSON-Struktur ungültig wird

```
{"invoiceNumber": "202158756", "invoiceDate": "2021-04-28", "invoiceAmount": 23.09} ] }
```

Abbildung 6.16 invalides JSON

Diese Beispiele verdeutlichen, dass Modellantworten nicht nur in Markdown-Blöcke eingebettet sein können, sondern auch durch zusätzliche Arrays unvollständige Strukturen oder fehlerhafte Klammerungen problematisch werden.

Die zugrunde liegende Implementierung ist im Laufe der Entwicklung der Extraktionsstrategien sukzessive erweitert worden und bildet inzwischen ein robustes Handling unterschiedlichster Ausgabeformate ab. Bei optimierten Prompts treten die genannten Fälle seltener auf, lassen sich jedoch nicht vollständig vermeiden. Ein Vergleich der Modelle zeigt deutliche Unterschiede in der Formatkonformität. Kleinere Modelle wie Gemma 3 1B liefern nur in rund 90% der Fälle direkt valides JSON und knapp 10% müssen über Fallback-Mechanismen verarbeitet werden. Bei Gemma 3 4B steigt die Erfolgsquote bereits auf etwa 98%. Leistungsstärkere Modelle wie GPT oder Claude halten sich nahezu vollständig an die Spezifikationen und erreichten im Test eine Erfolgsquote von 100%. Diese Unterschiede sind in Abbildung 6.17 dargestellt.

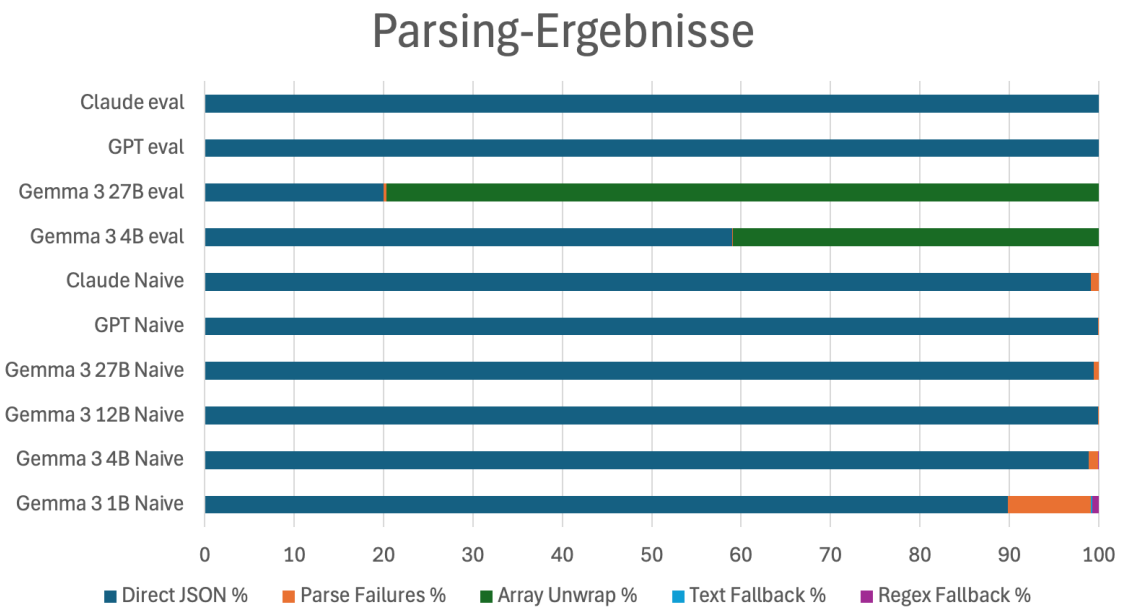


Abbildung 6.17 Verteilung der Parsing-Ergebnisse (detaillierte Werte siehe Tabelle A.3.1 im Anhang)

Der in der Klasse `BaseExtractionService` implementierte Parsing- und Normalisierungs-Prozess arbeitet mehrstufig. Abbildung 6.18 zeigt die zugrunde liegende Logik in Pseudocode-Form. Zunächst entfernt das System mögliche Markdown-Formatierungen und prüft den bereinigten Inhalt auf vollständige JSON-Strukturen. Liegt kein valides JSON vor durchsucht es die Antwort mithilfe regulärer Ausdrücke nach eingebetteten JSON-Blöcken und validiert diese einzeln. Schlägt auch dies fehl extrahiert das System als Fallback den Textbereich zwischen der ersten öffnenden und der letzten schließenden geschweiften Klammer und unterzieht diesen erneut einer JSON-Validierung

Sobald eine valide Struktur vorliegt entfernt das System gegebenenfalls unnötige äußere Arrays mit nur einem Element und normalisiert die extrahierten Daten. Dabei verarbeitet es verschachtelte Strukturen rekursiv entfernt Steuerzeichen und überführt Datums- und Zahlenangaben in ein einheitliches Format. Auf diese Weise wird sichergestellt, dass auch unvollständig formatierte oder fehlerhaft strukturierte Antworten konsistent weiterverarbeitet werden können.

```
function parse_llm_response(content):
    if content is empty:
        return error

    json_content = extract_json_from_response(content)
    if json_content is null:
        return as_text(content)

    parsed = parse_json(json_content)
    parsed = unwrap_single_array(parsed)
    return normalize_extracted_data(parsed)


function extract_json_from_response(content):
    remove_markdown_formatting(content)
    if is_valid_json(content):
        return content

    for pattern in json_patterns:
        for match in find_matches(content, pattern):
            if is_valid_json(match):
                return match

    extract_between_first_and_last_brace(content)
    if is_valid_json(extracted_part):
        return extracted_part

    return null


function normalize_extracted_data(data):
    recursively clean strings
    normalize dates & numbers
    return cleaned_data

```

Abbildung 6.18 Parsinglogik Pseudocode

7 Durchführung

Die Durchführung stellt die Brücke zwischen theoretischer Fundierung und praktischer Umsetzung dar. Sie zeigt, wie die in den vorangegangenen Kapiteln entwickelten Grundlagen in konkrete Experimente überführt werden und welche Erkenntnisse sich daraus gewinnen lassen. Im Zentrum steht die Untersuchung von LLM-basierten Extraktionsansätzen im Vergleich zu etablierten OCR-Verfahren. Durch die Anwendung auf einen umfangreichen und heterogenen Dokumentenkorpus werden Unterschiede in Genauigkeit, Robustheit und Effizienz sichtbar gemacht.

Ein wesentlicher Bestandteil ist die methodische Konstruktion eines belastbaren Evaluationsdatensatzes, der die Vielfalt realer Geschäftsdokumente widerspiegelt und damit als Grundlage für die Validität der Ergebnisse dient. Darauf aufbauend kommt die entwickelte Evaluationsplattform zum Einsatz, die eine konsistente Ausführung, Verwaltung und Auswertung der Experimente ermöglicht. Mit ihr lassen sich erste Leistungswerte erheben, die als Ausgangspunkt für die Beurteilung sowohl klassischer OCR-Ansätze als auch neuartiger LLM-Methoden dienen. Anschließend werden Optimierungsverfahren untersucht, die das Potenzial der LLM-Ansätze gezielt erweitern. Dazu gehören Variationen im Prompt Design, der Einsatz von Few-Shot-Beispielen sowie fortgeschrittene Strategien wie Multi-Modal Consensus und Multi-Agent-Orchestrierung. Auf diese Weise wird nachvollziehbar, wie sich durch schrittweise Anpassungen die Extraktionsqualität verbessern lässt und welche methodischen Implikationen sich daraus ergeben.

7.1 Testdatensatz-Engineering

Um den in Abschnitt 5.4 beschriebenen zweistufigen Stichprobenansatz umzusetzen, basiert die Evaluation nicht auf allen rund 400.000 verfügbaren Rechnungsdokumenten. Stattdessen werden zwei gezielt erstellte Teildatensätze verwendet, ein Developmentdatensatz mit 3.000 Rechnungen höherer Komplexität und ein Evaluationsdatensatz mit 10.000 zufällig gezogenen Dokumenten zur objektiven Leistungsbewertung.

Die Konstruktion des Developmentdatensatz folgt dem Prinzip einer gezielten Fehlerfokussierung. Ausgehend von einer Analyse der Ergebnisse des bestehenden Rechnungs-OCR werden Rechnungen mit extraktionskritischen Merkmalen ausgewählt. Um eine hohe Layoutvielfalt sicherzustellen, verteilt sich die Auswahl auf eine große Bandbreite von Kreditoren, was eine entsprechend breite Diversität an Layouts und Designs gewährleistet (vgl. Abschnitt 5.1).

Die Verwendung eines weiteren, vom Developmentdatensatz unabhängigen Evaluationsdatensatz dient der Validierung der entwickelten Verfahren auf einem breiteren und unvoreingenommenen Datensatz. Diese Trennung stellt sicher, dass die Strategien nicht implizit an bereits bekannten Beispielen getestet werden und die finale Bewertung den Charakter einer unabhängigen Validierung behält.

Die Erstellung der Datensätze wird im Folgenden entlang der Auswahl- und Filterkriterien dargestellt. Die Darstellung erfolgt in konsolidierter Form, um die Nachvollziehbarkeit des Vorgehens zu gewährleisten.

7.1.1 Rechnungskomplexität

Die Fehlerfokussierung basiert auf der Diskrepanz zwischen Rechnungs-OCR und Infact-Referenzdaten als Indikator für Extraktionsschwierigkeit. Dokumente mit Abweichungen zwischen diesen beiden Systemen weisen typischerweise komplexere Layouts, ungewöhnliche Formatierungen oder andere Faktoren auf, die das bestehende Extraktionsverfahren vor Herausforderungen stellen (vgl. Abschnitt 8.3.1).

Für die diskrepanzbasierte Klassifikation kommt eine binäre Bewertung zum Einsatz. Dokumente mit identischen Ergebnissen zwischen Rechnungs-OCR und Infact werden als „GOOD" klassifiziert, solche mit Abweichungen als „BAD". Diese Kategorisierung bildet die Grundlage für die gezielte Auswahl schwieriger Fälle im Developmentdatensatz.

7.1.2 Vorbereitung und Datenbereinigung

Die Aufbereitung der ursprünglich 767.817 Rechnungsdokumente in evaluationstaugliche Datensätze erfolgt durch die Erstellung der Materialized View *prediction_results_polished*, die eine Filterung und Bereinigung der Daten beinhaltet. Diese View kombiniert und normalisiert die relevanten Rechnungs-OCR-Ergebnisse und Infact-Referenzdaten und ermöglicht dadurch eine effiziente Abfrage und Analyse. Die in Abschnitt 7.1.1 eingeführte GOOD/BAD-Klassifikation wird wie in Abbildung 7.1 dargestellt umgesetzt.

```
 1. CREATE MATERIALIZED VIEW prediction_results_polished AS
 2. SELECT documents.id,
 3.        e_t.content -> 'text'::TEXT                AS text,
 4.        (e_g.content -> 'invoiceNumber')::TEXT     AS gini_invoiceNumber,
 5.        (e_i.content -> 'invoiceNumber')::TEXT     AS infact_invoiceNumber,
 6.        -- Feldvergleiche und Diskrepanzermitlung
 7.        CASE WHEN (e_g.content -> 'invoiceNumber')::text =
 8.                  (e_i.content -> 'invoiceNumber')::text
 9.             THEN 'GOOD' ELSE 'BAD' END            AS number_prediction,
10.        -- ...
11.        CASE WHEN (e_g.content ->> 'invoiceAmount') ~ '^\d+(\.\d+)?$'
12.                AND (e_i.content ->> 'invoiceAmount') ~ '^\d+(\.\d+)?$'
13.                AND (e_g.content ->> 'invoiceAmount')::TEXT != 'null'
14.                AND (e_g.content ->> 'invoiceAmount')::TEXT::FLOAT::TEXT =
15.                    (e_i.content ->> 'invoiceAmount')::TEXT::FLOAT::TEXT
16.             THEN 'GOOD' ELSE 'BAD' END            AS amount_prediction,
17.        -- ...
18.        CASE WHEN (e_g.content -> 'invoiceDate')::text = (e_i.content -> 'invoiceDa-
te')::text
19.             THEN 'GOOD' ELSE 'BAD' END AS date_prediction,
```

Abbildung 7.1 Schematische Darstellung der GOOD/BAD-Klassifikation (vollständiges SQL-Query im Anhang 2.2)

Die Diskrepanzermittlung erfolgt feldspezifisch mit entsprechenden Vergleichslogiken. Während Rechnungsnummern und Datumsangaben durch direkte String-Vergleiche bewertet werden, erfordert die Betragsvalidierung zusätzliche numerische Prüfungen. Das Regex-Pattern `^\d+(\.\d+)?$` stellt sicher, dass nur numerische Inhalte verarbeitet werden. Diese Validierung ist technisch zwingend erforderlich, da andernfalls ein Typenumwandlung von nicht-numerischen Werten zu einem SQL-Fehler führen würde. Die Typenumwandlung (`::FLOAT::TEXT`) sorgt zudem für eine einheitliche Darstellung (z. B. „123.40" → „123.4") und verhindert Vergleichsfehler. Diese Validierungslogik wurde in Vorversuchen empirisch geprüft und erwies sich als zuverlässig, da fehlerhafte oder inkonsistente Feldwerte so effizient ausgeschlossen werden können.

Die Entfernung nicht evaluierbarer Dokumente basiert, wie in Abbildung 7.2 dargestellt, auf den zuvor extrahierten Texten aus den PDF-Dateien:

```
        -- Weitere Feldvergleiche...
FROM documents
JOIN extractions e_t ON documents.id = e_t.document_id
     AND e_t.extraction_type = 'EXTRACT_TEXT'
     AND length((e_t.content -> 'text')::TEXT) > 25
     AND (e_t.content -> 'text')::TEXT NOT LIKE '(cid:%'
     AND (e_t.content -> 'text')::TEXT NOT LIKE '%(cid:%'
     AND (length((e_t.content -> 'text')::TEXT) -
          length(replace((e_t.content -> 'text')::TEXT, '\ufffd', ''))) < 30
     AND (e_t.content -> 'text')::TEXT NOT LIKE '%tundenzettel%'
     AND ((e_t.content -> 'text'::text)::text) !~~ '%Negatives Abrechnungskont%'::text
     AND length((e_t.content -> 'text')::TEXT) < 10000
```

Abbildung 7.2 Schematische Darstellung der Filterung nicht evaluierbarer Dokumente (vollständiges SQL-Query im Anhang 2.2)

Die Bereinigung beginnt mit dem Ausschluss unvollständiger Daten. Dokumente, bei denen entweder Gini- oder Infact-Ergebnisse fehlen, werden nicht berücksichtigt. Ebenfalls entfernt werden Fälle mit fehlendem Absendernamen oder einem Absender, der mit „aifinyo" beginnt, um interne Test- und Beispieldokumente auszusortieren.

Im nächsten Schritt werden technische Filter angewendet. Dazu gehört der Ausschluss sehr kurzer Texte unter 25 Zeichen, die meist auf bildbasierte Inhalte hinweisen und sich nicht sinnvoll textbasiert verarbeiten lassen. Sehr lange Texte über 10.000 Zeichen werden ebenfalls ausgeschlossen, um unnötig große Konteingaben zu vermeiden, die die Verarbeitung verlangsamen und bei lokalen Modellen wie Gemma 3 Hardwaregrenzen ausreizt. Weitere technische Filter schließen Dokumente mit typischen Rendering-Problemen wie CID-Font-Artefakten oder einer hohen Anzahl an Unicode-Ersatzzeichen (�) aus, da diese in Tests gar nicht oder nur unzureichend extrahiert werden können. Ergänzend kommen semantische Filter zum Einsatz, die Dokumente mit für die Evaluation irrelevanten Inhalten aus dem Datenbestand entfernen. Dazu zählen etwa Stundenzettel oder Abrechnungen des negativen Abrechnungskontos, die im Geschäftsprozess der aifinyo AG einen Sonderfall darstellen. Die gewählten Grenzwerte (z. B. minimale Textlänge von 25 Zeichen, maximale Textlänge von 10.000 Zeichen, Unicode-Schwelle von 30 Ersatzzeichen) basieren auf Erfahrungswerten aus Vorversuchen und dienen dazu, eine Balance zwischen Datenqualität und Datenquantität sicherzustellen.

Die Auswirkungen dieser Filterungen auf den Gesamtdatenbestand mit erfolgreicher Textextraktion sind in Tabelle 7.1 dargestellt.

Filterkategorie	**Anzahl Dokumente**	**Anteil am Gesamtdatenbestand**
Fehlende Gini- oder Infact-Ergebnisse	316 943	41,28%
Fehlender Absendername, Absender beginnt mit aifinyo	2 420	0,54%
Textlänge kleiner als 25 Zeichen	34 670	7,73%
Textlänge größer als 10.000 Zeichen	8 208	1,83%
CID-Font-Artefakte, viele Unicode-Ersatzzeichen	12 085	2,69%
Stundenzettel, negatives Abrechnungskonto	14 836	3,31%
Gesamt entfernt	68 598	15,30%
Verbleibend	379 856	84,70%

Tabelle 7.1 Auswirkungen der Filterung auf den Gesamtdatenbestand

Nach Abschluss dieser Bereinigung verbleibt eine qualitätsgesicherte Basis von 379.856 evaluationstauglichen Dokumenten mit klarer GOOD/BAD-Klassifikation je zu untersuchendem Extraktionsfeld.

7.1.3 Evaluationsdatensatz

Der Evaluationsdatensatz wird zufällig aus den Daten der `prediction_results_polished`-View gezogen und unterliegt keiner weiteren Verzerrung durch zusätz-liche Filter oder Komplexitätsbewertungen. Die Erstellung erfolgt vor dem Developmentdatensatz, wodurch sichergestellt ist, dass keine Überschneidungen oder Duplikate zwischen den beiden Sets auftreten.

Die Auswahl erfolgt einmalig per RANDOM()-Sortierung (Abbildung 7.3 Materialized View zur Erstellung des) und wird in einer Materialized View persistiert:

```
1. CREATE MATERIALIZED VIEW evaluationsset AS
2. SELECT * FROM prediction_results_polished
3. ORDER BY RANDOM() LIMIT 10000;
```

Abbildung 7.3 Materialized View zur Erstellung des Evaluationsdatensatzes

Durch die Persistierung in einer Materialized View bleibt die Zusammensetzung des Sets über alle Experimente hinweg konstant. Dieses Vorgehen verhindert unbeabsichtigte Datenverschiebungen zwischen Evaluationsphasen und gewährleistet reproduzierbare Vergleichsbedingungen.

Der Abgleich der Extraktionsmetriken mit dem Gesamtdatenbestand bestätigt die Repräsentativität des 10.000-Dokumente-Samples (Tabelle 7.2 Vergleich der Extraktionsmetriken zwischen Gesamtdatenbestand und). Zusätzlich deckt der Evaluationsdatensatz 508 unterschiedliche Kreditoren ab, wodurch eine breite Layoutvielfalt abgebildet wird. Damit wird sichergestellt, dass das Set nicht nur hinsichtlich der Metriken, sondern auch strukturell repräsentativ für den Gesamtdatenbestand ist.

Set	***Overall Acc***	***Doc Acc***	***Number Acc***	***Date Acc***	***Amount Acc***
Evaluations-datensatz	95,53%	87,24%	96,72%	93,37%	96,51%
All	95,1%	86,3%	96,4%	92,6%	96,2%

Tabelle 7.2 Vergleich der Extraktionsmetriken zwischen Gesamtdatenbestand und Evaluationsdatensatz

Die geringen Abweichungen zwischen Evaluationsdatensatz und Gesamtdatenbestand belegen eine hinreichende Repräsentativität. Für die Bewertung der Extraktionsleistung kommen sowohl feldspezifische Genauigkeiten als auch dokumentbasierte Metriken zum Einsatz.

Die Bewertung erfolgt anhand der in Abschnitt 5.3.2 definierten Extraktions-Metriken. Insbesondere die $Document\ Accuracy$ stellt einen strengen Maßstab dar, da bereits ein einzelner Fehler in Rechnungsnummer, Datum oder Betrag zur Abwertung des gesamten Dokuments führt. Während die Einzelfelder Genauigkeiten zwischen 93% und 97% erreichen, liegt die $Document\ Accuracy$ bei nur 87,24%, da sich Fehlerwahrscheinlichkeiten kumulieren. Diese Werte bilden die Grundlage für die spätere Bewertung der LLM-Modelle.

7.1.4 Developmentdatensatz

Das gezielte Leistungsprofil des Developmentdatensatz zeigt sich deutlich im Vergleich der Extraktionsmetriken (Tabelle 7.2).

Set	***Overall Acc***	***Document Acc***	***Number Acc***	***Date Acc***	***Amount Acc***
Development-datensatz	84,01%	57,87%	85,70%	83,17%	83,17%
All	95,1%	86,3%	96,4%	92,6%	96,2%

Tabelle 7.3 Vergleich der Extraktionsmetriken zwischen Gesamtdatenbestand und Developmentdatensatz

Der Developmentdatensatz wird über die Materialized View `developmentset` implementiert. Diese realisiert eine SQL-basierte Auswahl mit Komplexitätsfokus und gewährleistet eine hohe Layout-Diversität durch möglichst gleichverteilte Kreditorenauswahl anstelle einer rein zufälligen Ziehung. Hintergrund ist die Beobachtung, dass eine große Anzahl unterschiedlicher Kreditoren in der Regel auch eine größere Vielfalt an Layouts, Tabellenstrukturen, Textanordnungen und Designelementen mit sich bringt (Abbildung 7.4).

```
CREATE MATERIALIZED VIEW developmentset AS
-- Ausschluss des Evaluationssets
WITH eval_set_ids AS (
    SELECT id FROM evaluationsset
),
base AS (
    SELECT *, ROW_NUMBER() OVER () AS internal_order
    FROM prediction_results_polished
    WHERE id NOT IN (SELECT id FROM eval_set_ids)
),
-- Fehlerbasierte Label-Gruppen
number_good AS (SELECT id, 'number_good' AS label, internal_order FROM base WHERE number_prediction = 'GOOD'),
     number_bad AS (SELECT id, 'number_bad' AS label, internal_order FROM base WHERE number_prediction = 'BAD'),
     amount_good AS (SELECT id, 'amount_good' AS label, internal_order FROM base WHERE amount_prediction = 'GOOD'),
     amount_bad AS (SELECT id, 'amount_bad' AS label, internal_order FROM base WHERE amount_prediction = 'BAD'),
-- Gezielte Diversitätsstrategie
sender_diverse AS
(SELECT *, ROW_NUMBER() OVER
 (PARTITION BY label, sender_prediction
  ORDER BY internal_order) AS sender_rank
 FROM joined_with_sender WHERE sender_rank = 1
),
-- Fallback-Auffüllung
fallback_fill AS
(SELECT *
 FROM (SELECT *, ROW_NUMBER() OVER
       (PARTITION BY label
        ORDER BY internal_order) AS group_rank
       FROM joined_with_sender
       WHERE id NOT IN (SELECT id FROM sender_diverse)) t
 WHERE group_rank <= 500)
```

Abbildung 7.4 Ausschnitte der Materialized View zur Erstellung des Developmentdatensatzes (vollständiges SQL Querry im Anhang 2.3)

Die Erstellung beginnt mit dem Ausschluss aller Dokumente, die im Evaluationsset enthalten sind. Danach erfolgt eine feldspezifische Kategorisierung nach Extraktionserfolg (GOOD/BAD) für Rechnungsnummer, Betrag und Datum. Jedes Dokument erhält ein eindeutiges Label, beispielsweise number_bad oder amount_good, das als Hilfskategorie für die nachfolgende Diversitätslogik dient. Tritt ein Dokument in mehreren Kategorien auf, bestimmt eine Priorisierungslogik (ROW_NUMBER) die Zuordnung.

Zur Maximierung der strukturellen Vielfalt wird pro Fehlerkategorie und Kreditor jeweils ein Dokument ausgewählt. Bleiben pro Kategorie weniger als 500 Dokumente übrig, ergänzt ein Fallback-Verfahren die Auswahl. Die finale Auswahl kombiniert diese strukturierte Vorauswahl mit einem kleineren Anteil zufälliger Ergänzungen und begrenzt den Bestand anschließend auf exakt 3.000 Dokumente.

Der Developmentdatensatz enthält bewusst einen höheren Anteil schwieriger Fälle (ca. 42%) als der Gesamtdatenbestand (14%). Die Schwierigkeit bezieht sich dabei auf die in Abschnitt 5.3.2 definierte $Document\ Accuracy$, also den strengen Maßstab, nach dem bereits ein einzelner Fehler in Rechnungsnummer, Datum oder Betrag zur Abwertung des gesamten Dokuments führt. Zusätzlich deckt das Set 333 unterschiedliche Kreditoren ab, was die hohe Layoutvielfalt und strukturelle Diversität eindrucksvoll belegt. Diese Segmentierung kombiniert eine ausgeprägte Fehlerdiversität mit einer breiten Kreditorenbasis und schafft damit optimale Bedingungen für die iterative Entwicklung und Optimierung der LLM-basierten Extraktionsstrategien.

7.2 Initiale Tests

Nach der Implementierung der Evaluationsplattform beginnt die Untersuchung der ersten Extraktionsstrategie. Ziel dieser initialen Testphase ist nicht nur die Bewertung der reinen Modellleistung, sondern auch die Identifikation potenzieller Fehlerquellen in der Datenbasis und den vorgelagerten Verarbeitungsschritten. Zur Sicherstellung der Vergleichbarkeit werden in allen Modellen identische Prompts eingesetzt, wodurch eine faire Bewertung der Ausgangsleistung ohne modellspezifische Optimierungen möglich ist.

7.2.1 Ausgangs-Performance

Die ersten Testergebnisse liefern anhand eines einheitlichen Prompts einen direkten Vergleich zwischen den eingesetzten LLMs und dem bisherigen Rechnungs-OCR von Gini. Die initiale Performance-Messung erfolgt zunächst auf einer noch nicht vollständig bereinigten Datenbasis und dient der Identifikation potenzieller Fehlerquellen (Abschnitt 7.3). Für die in Abbildung 7.6 dargestellten Ergebnisse wird jedoch die final bereinigte und nachkorrigierte Datenbasis verwendet, um eine konsistente Vergleichbarkeit sicherzustellen.

Ausgangs-Prompt:

Das in Abbildung 7.5 gezeigte Baseline-Prompt folgt einem klassischen Zero-Shot-Setup und fordert die Extraktion aller Basisinformationen in einem JSON-Format an, ohne zusätzliche Beispiele oder weiterführende Instruktionen. Diese bewusst schlanke Form macht die reine Ausgangsleistung der Modelle sichtbar, bevor gezielte Optimierungen eingeführt werden.

```
System:
You are an AI that extracts invoice information and returns it as valid JSON.
- Do NOT include markdown formatting.
- Respond with raw JSON only.
- Dates (invoiceDate) must be ISO date strings (YYYY-MM-DD).
- Validate datatypes strictly.
- ...

User:
Extract the following invoice details and return ONLY based on this JSON structure:
{ "invoiceNumber": "String", "invoiceDate": "Date", "invoiceAmount": "Double", "sender": {
"name": ... }
Process the following extracted text and return ONLY the JSON: $text
```

Abbildung 7.5 Baseline-Prompt – gekürzter Ausschnitt (vollständige Version im Anhang 1.2)

Die Resultate in Abbildung 7.6 zeigen, dass GPT-4.1, Claude 3 Sonnet und Gemma 3 27B-IT die besten Ergebnisse erzielen, mit rund 97% $Overall\ Accuracy$ und über 93% $Document\ Accuracy$. Besonders bemerkenswert ist, dass Gemma 3 27B-IT als offenes Modell nahezu gleichauf mit den Closed-Source-Systemen liegt. Dies deckt sich mit den aktuellen Benchmarks vom 8. März 2025, in denen Gemma 3 27B-IT mit einem Elo-Score von 1339 in den Top 10 der LMSys Chatbot Arena geführt wird und sich dort mit Closed-Source-Modellen wie Claude 3 und GPT-4 messen kann [93]. Gemma 3 12B-IT liegt mit 96,04% $Accuracy$ und 89,47% $Document\ Accuracy$ etwas darunter. Deutlich schwächer schneiden kleinere Modelle wie Gemma 3 4B-IT (82,68% / 63,03%) und Gemma 3 1B-IT (54,25% / 24,97%) ab. Als historische Vergleichsbasis dient Gini mit 84,01% $Accuracy$ und 57,87% $Document\ Accuracy$.

Alle Tests erfolgen unter identischen Parametereinstellungen, die in Tabelle 7.4 zusammengefasst sind.

	$temperature$	top_p	max_tokens
Parameter	0	1	8192

Tabelle 7.4 Parameter-Setup der Ausgangs-Performance-Experimente

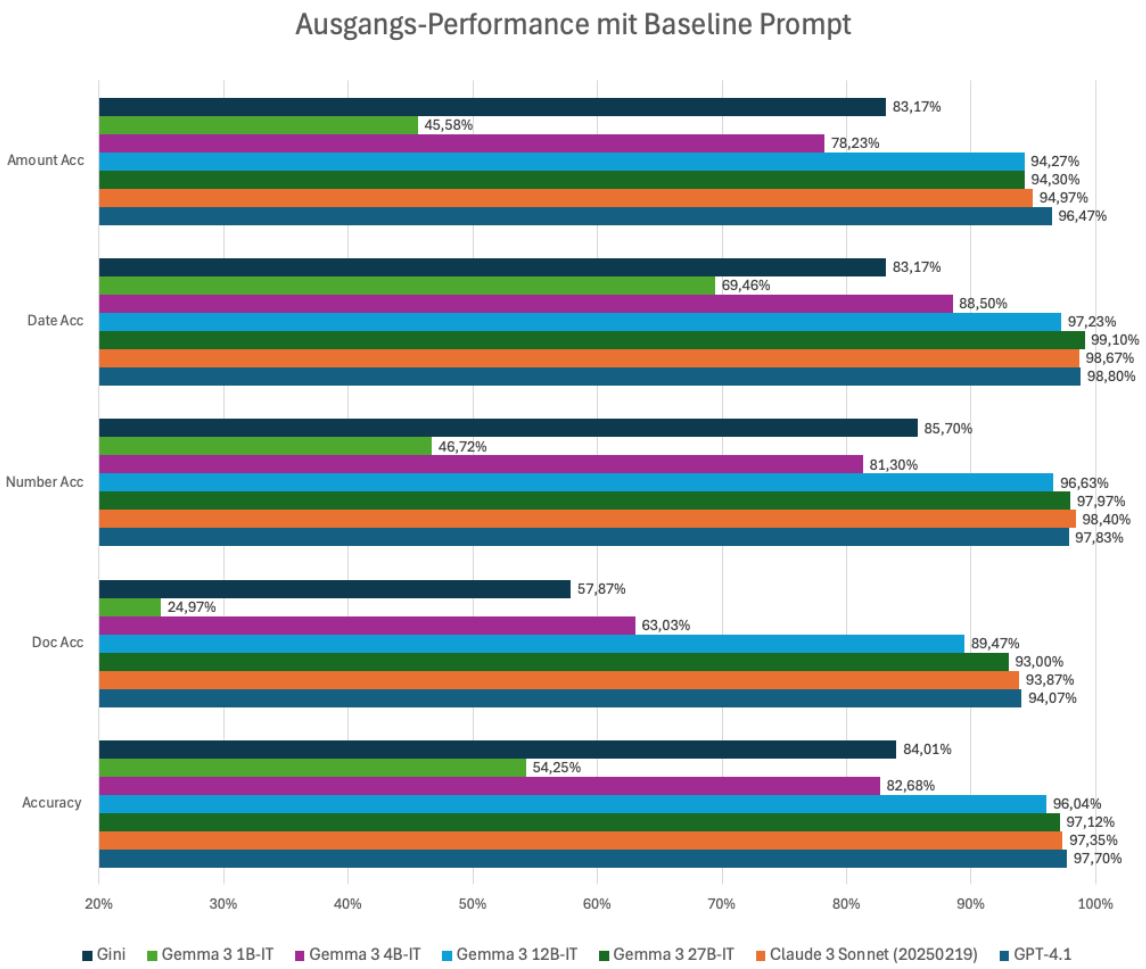


Abbildung 7.6 Ausgangs-Performance mit Baseline Prompt (vollständige Werte im Anhang 3.2)

Aus den ersten Tests ergibt sich eine zentrale Anpassung für die weiteren Experimente. Die Abfrage ist auf die Felder $invoiceNumber$, $invoiceDate$ und $invoiceAmount$ beschränkt (Abbildung 7.7), da diese Felder am besten annotiert und am zuverlässigsten validierbar sind. Gleichzeitig führt die Reduktion zu einer deutlich geringeren Anzahl generierter Completion-Tokens (Tokens, die das Modell als Antwort auf den Prompt produziert), was die durchschnittliche Verarbeitungszeit pro Dokument erheblich verkürzt. Je nach Modell sinkt die Dauer im Mittel um 88% bis 97%, von teils über einer Minute auf durchschnittlich 1,73 s bei Gemma 3 27B-IT bzw. 0,45 s bei Gemma 3 4B-IT (Abbildung 7.8). Darüber hinaus zeigt sich eine Verbesserung der Extraktionsgenauigkeit, insbesondere bei Gemma 3 4B-IT mit einem Anstieg von 82,68% auf 91,12%.

```
User:
Extract the following invoice details and return ONLY based on this JSON structure:
{ "invoiceNumber": "String", "invoiceDate": "Date", "invoiceAmount": "Double" }
Process the following extracted text and return ONLY the JSON: $text
```

Abbildung 7.7 Ausschnitt verkürzter Prompt (vollständige Version im Anhang 1.3)

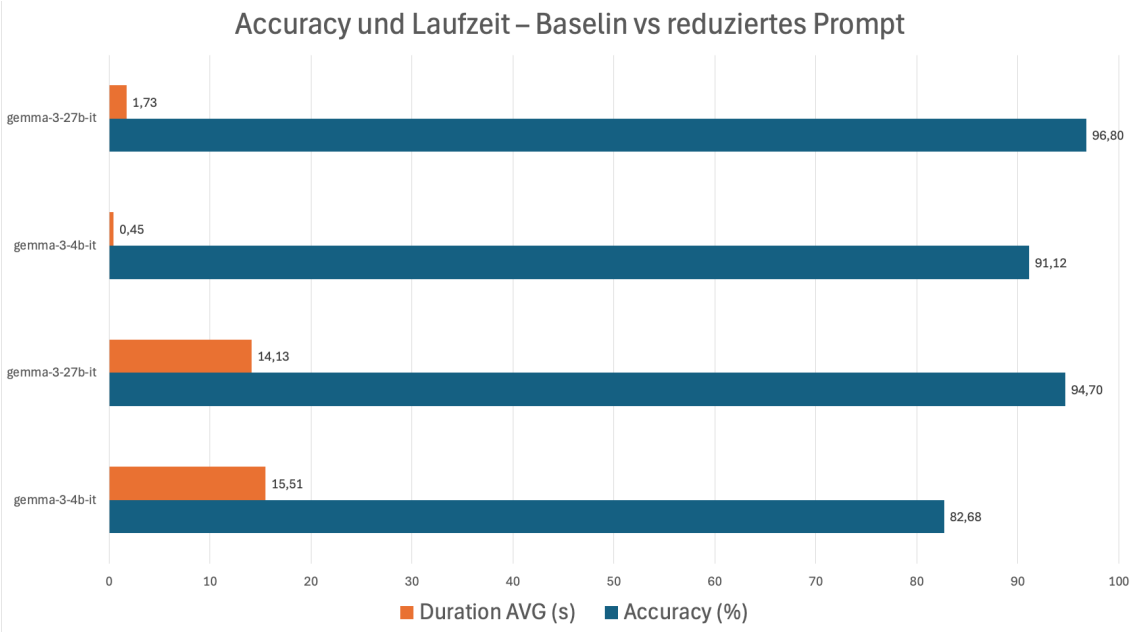


Abbildung 7.8 $Overall\ Accuracy$ und Laufzeit – Baseline vs reduzierter Prompt (vollständige Version im Anhang 3.3)

7.2.2 Einfluss von Parametervariationen auf die Modellleistung

Nach der Bestimmung der Ausgangsleistung wird der Einfluss verschiedener Modellparameter auf Extraktionsqualität, Reproduzierbarkeit und Laufzeit untersucht. Zum Einsatz kommt Gemma 3 4B-IT, das ein günstiges Verhältnis von Genauigkeit und Inferenzzeit für explorative Experimente bietet.

Im Fokus stehen die Parameter $temperature$, top_p und max_tokens. Alle Konfigurationen nutzen den verkürzten Prompt (Abbildung 7.7) und werden im Zero-Shot-Setup ausgeführt. Jede Variante wird viermal auf demselben Datensatz mit 3.000 Rechnungen getestet, um Schwankungen durch Nichtdeterminismus zu erfassen. Ein fester Seed (3333) sorgt dafür, dass verbleibende Unterschiede auf numerische Einflüsse beim Inferenzprozess zurückzuführen sind.

Die getesteten Konfigurationen sind in Tabelle 7.5 dargestellt:

Strategie	$\boldsymbol{temperature}$	$\boldsymbol{top-p}$	$\boldsymbol{max-tokens}$
Temp 0	0	0	200
Temp 0.1	0.1	0	200
Top-P 1	0	1	200
Both 1	1	1	200
Max Tokens 50	0	0	50

Tabelle 7.5 Getestete Parameterkonfigurationen für Gemma 3 4B-IT

Die drei Konfigurationen Temp 0, Top-P 1 und Temp 0.1 zeigen eine insgesamt hohe Reproduzierbarkeit. Bei Temp 0 sowie Top-P 1 liegen die $Overall - Accuracy$-Werte stabil zwischen 91,12% und 91,23%, die $Document\ Accuracy$ zwischen 77,07% und 77,33%. Auch bei Temp 0.1 bleiben die Ergebnisse nahezu konstant, mit maximalen Schwankungen von 0,06 Prozentpunkten $Overall\ Accuracy$ und 0,14 Prozentpunkten $Document\ Accuracy$). Solche minimalen Unterschiede werden in der Literatur als typische Rundungseffekte bei GPU-Inferenz beschrieben [94].

Besonders auffällig ist die vollständige Reproduzierbarkeit bei Both 1. In allen vier Durchläufen bleiben $Overall\ Accuracy$ und $Document\ Accuracy$ exakt konstant (91,12% bzw. 77,07%). Dieser perfekte Determinismus erklärt sich durch den verwendeten festen Seed (3333), der auch bei stochastischen Sampling-Parametern reproduzierbare Ergebnisse gewährleisten soll. Die minimalen Schwankungen bei den anderen Konfigurationen (Temp 0, Temp 0.1, Top-P 1) sind daher umso bemerkenswerter, da diese trotz theoretisch deterministischer bzw. minimal stochastischer Parameter nicht vollständig reproduzierbar sind. Dies deutet auf implementierungsspezifische Unterschiede in der Seed-Kontrolle bei verschiedenen Sampling-Strategien in LM Studio hin.

Zudem zeigen empirische Studien, dass Änderungen der Sampling-Temperatur im Bereich von 0.0 bis 1.0 keine signifikanten Unterschiede in Problem-Solving-Aufgaben erzeugen [65]. Während sich diese Untersuchung auf Multiple-Choice-Reasoning konzentriert, legen die hier erzielten Ergebnisse nahe, dass auch bei strukturierter Informationsextraktion moderate Parameteränderungen wenig Einfluss haben.

Die einzige Konfiguration mit klarer Verschlechterung ist die Reduktion des max_tokens-Limits auf 50. Hier liegt die $Overall\ Accuracy$ konstant bei 5,53% bis 5,55%, die $Document\ Accuracy$ zwischen 4,90% und 4,93%. Ursache ist das Abschneiden vieler JSON-Antworten, die dadurch nicht mehr parsbar sind. Dieses Ergebnis verdeutlicht die Notwendigkeit, das max_tokens-Limit konservativ an der erwarteten Antwortgröße auszurichten.

Die Zusammenfassung der Mittelwerte und Spannweiten ist in Tabelle 7.6 Einfluss von Parametervariationen auf die Modellleistung bei gemma-3-4b-it (jeweils Durchschnitt aus vier Läufen) dargestellt.

Konfiguration	$\boldsymbol{Overall\ Accuracy\ (AVG)}$	**Spannweite**	$\boldsymbol{Document\ Accuracy\ (AVG)}$	**Spannweite**
Temp 0	91,18%	91,12% – 91,23 %	77,20%	77,07% – 77,33%
Temp 0.1	91,21%	91,19% – 91,25%	77,28%	77,10% – 77,29%
Top-P 1	91,19%	91,14% – 91,25%	77,22%	77,10% – 77,29%
Both 1	91,12%	91,12% – 91,12%	77,07%	77,07% – 77,07%
Max Tokens 50	5,54%	5,53% – 5,55%	4,91%	4,90% – 4,93%

Tabelle 7.6 Einfluss von Parametervariationen auf die Modellleistung bei gemma-3-4b-it (jeweils Durchschnitt aus vier Läufen)

Zusätzlich zeigt sich, dass einzelne Antworten trotz reduzierter Zielstruktur (`invoiceNumber`, `invoiceDate`, `invoiceAmount`) bis zu 199 Tokens lang sind, bei einem Durchschnitt von 54 Tokens. Ursache ist, dass das Modell teilweise Arrays mit alternativen JSON-Formaten erzeugt. In diesen Fällen wird stets das erste JSON-Objekt ausgewertet. Dies verdeutlicht die Bedeutung einer gut gewählten max_tokens-Obergrenze. Ein zu niedriger Wert führt zu abgeschnittenen Antworten, ein zu hoher kann unnötige Alternativen und höhere Laufzeiten erzeugen. Der hier gewählte Wert von 200 stellt einen angebrachten Kompromiss dar.

Für die weiteren Experimente wird bei Gemma 3 eine Sampling-Konfiguration mit $temperature = 1$, $top\_p = 1$ und fixiertem Seed eingesetzt, da sie deterministische Ergebnisse liefert. Proprietäre Modelle wie GPT-4 und Claude 3 werden hingegen mit $temperature = 0$, $top\_p = 1$ betrieben. Die finalen Parameter sind in Tabelle 7.7 zusammengefasst.

Die final verwendeten Parameter sind in Tabelle 7.7 zusammengefasst.

Modelltyp	$\boldsymbol{temperature}$	$\boldsymbol{top-p}$	$\boldsymbol{max-tokens}$	$\boldsymbol{Seed}$
Claude 3 Sonnet	0	1	200	–
GPT-4.1	0	1	200	–
Gemma 3 (alle Größen)	1	1	200	3333

Tabelle 7.7 Finale Basis Parameter

7.2.3 Einfluss der Textextraktion auf die Modellleistung

Die Qualität der Textextraktion aus PDF-Dokumenten beeinflusst unmittelbar die Modellleistung, da alle Extraktionsstrategien auf dem extrahierten Text basieren. Unterschiede ergeben sich durch Fehler in der Zeichenerkennung, Variationen in der Segmentierung, den Umgang mit Zeilenumbrüchen sowie Abweichungen in Zeichencodierung und Interpunktion. Besonders formatabhängige Felder wie Rechnungsnummern und Beträge reagieren empfindlich auf solche Unterschiede.

Zur Untersuchung dieses Einflusses werden die in Abschnitt 6.3.3 eingeführten Textextraktionsbibliotheken getestet. Die Tests erfolgen mit Gemma 3 27B-IT im Zero-Shot-Setup mit standardisiertem Prompt auf denselben 3.000 Rechnungen. Die Ergebnisse sind in Tabelle 7.8 dargestellt.

Bibliothek	$\boldsymbol{Overall\ Acc}$	$\boldsymbol{Document\ Acc}$	$\boldsymbol{Number\ Acc}$	$\boldsymbol{Date\ Acc}$	$\boldsymbol{Amount\ Acc}$
pdfplumber	97,32%	93,20%	98,27%	99,37%	94,33%
pdfminer	97,12%	92,80%	97,47%	98,53%	95,37%
PyMuPDF	96,99%	92,43%	97,43%	98,90%	94,63%
pdf_reader (Ruby)	96,63%	92,00%	97,70%	99,33%	93,77%
pypdfium2	93,05%	89,43%	93,43%	94,73%	90,99%
Tesseract OCR	95,24%	88,26%	94,36%	98,23%	93,13%

Tabelle 7.8 Einfluss der Textextraktionsbibliothek auf die Extraktionsgenauigkeit; Gemma 3 27B-IT, 3.000 Rechnungen

Die Ergebnisse zeigen ein insgesamt hohes Leistungsniveau der parserbasierten Lösungen. pdfplumber schneidet dabei in fast allen Kategorien am besten ab. Mit 97,32% $Accuracy$ und 93,20% $Document\ Accuracy$ erzielt es die höchsten Gesamtwerte und liegt auch bei Rechnungsnummern (98,27%) sowie Datumsangaben (99,37%) vorne. Lediglich im Feld `invoiceAmount` erreicht pdfminer mit 95,37% den besten Einzelwert, während pdfplumber hier 94,33% erzielt.

PyMuPDF und pdf_reader (Ruby) liefern vergleichbare Resultate, bleiben aber in der $Document\ Accuracy$ leicht hinter pdfplumber und pdfminer zurück. Die Ergebnisse von pypdfium2 fallen mit 93,05% $Accuracy$ und 89,43% $Document\ Accuracy$ deutlich niedriger aus. Da der Fokus dieser Arbeit auf der Evaluation verschiedener LLM-Strategien liegt, wird auf eine vertiefte Analyse der bibliotheksspezifischen Unterschiede verzichtet. Tesseract OCR, als rein bildbasierte Methode, fällt mit 88,26% $Document\ Accuracy$ klar hinter die parserbasierten Verfahren zurück. Die Ergebnisse lassen sich insbesondere auf charakteristische OCR-bedingte Fehler, wie die Verwechslung ähnlich aussehender Zeichen („0" und „O", „1" und „I") zurückführen.

Auf Grundlage der Ergebnisse wird pdfplumber als Standardbibliothek für die weiteren Experimente festgelegt, da es die höchsten Genauigkeitswerte erzielt.

7.3 Einfluss der Ground-Truth-Qualität

Eine zentrale Erkenntnis der ersten Tests ist, dass die als Ground Truth verwendeten Infact-Daten nicht die erwartete Qualität aufweisen. Abweichungen zwischen den LLM-Extraktionen und den Referenzdaten führen zu fälschlicherweise negativen Bewertungen der Modellleistung, da die LLMs in einigen Fällen korrekte Ergebnisse liefern, während die Referenzdaten geschäftsbedingt modifiziert sind oder tatsächliche Fehler enthalten.

Die Analyse der Abweichungen zeigt vor allem folgende Ursachen:

- **Rechnungsdatum:** Anpassungen durch die Kundenbetreuung, um Mahnungen zu verzögern (Mahnaufschub). Mangels einer dedizierten Funktion wird das Rechnungsdatum manuell nach hinten verschoben.
- **Rechnungsbetrag:** In seltenen Fällen angepasste Werte durch Workarounds, etwa bei Verrechnungen oder Kulanzregelungen.
- **Rechnungsnummer:** Zusätze wie „INK" zur internen Kennzeichnung von Inkassofällen.
- **Menschliche Fehler:** Teilweise Übernahme fehlerhafter OCR-Ergebnisse, z. B. Verwechslung von „0" und „O" oder fehlende Leerzeichen.

Zur Sicherstellung einer fairen Evaluationsbasis werden die betroffenen Datensätze manuell nachkorrigiert. Die Korrektur erfolgt durch Sichtung aller fehlerhaften Extraktionen und manuelle Nachprüfung über die implementierte Ground-Truth-Korrekturfunktionalität. Korrekte Ergebnisse werden nicht erneut überprüft, sodass ein Restrisiko vereinzelter verbliebener Fehler besteht. Insgesamt sind 6% der Dokumente (180 von 3.000), oder 2,13% aller Werte betroffen. Alle Anpassungen werden transparent dokumentiert.

Die Auswirkungen der Korrekturen auf den Developmentdatensatz sind in Tabelle 7.9 dargestellt. Die $Overall\ Accuracy$ steigt durch die Bereinigung von 83,13% auf 84,03%, die $Document\ Accuracy$ verbessert sich von 56,07% auf 57,87%. Besonders deutlich wirkt sich die Bereinigung beim Rechnungsdatum und beim Rechnungsbetrag aus, deren $Accuracy$ um knapp einen Prozentpunkt steigt. Ohne die Korrekturen lägen die ausgewiesenen Genauigkeiten von Rechnungs-OCR und LLMs systematisch unter ihrem tatsächlichen Niveau.

Feld	**vor Korrektur**	**nach Korrektur**	**Veränderung**
$Number\ Acc$	84,73%	85,70%	+0,97%
$Date\ Acc$	82,30%	83,23%	+0,93%
$Amount\ Acc$	82,37%	83,17%	+0,80%
$Overall\ Acc$	83,13%	84,03%	+0,90%
$Document\ Acc$	56,07%	57,87%	+1,80%

Tabelle 7.9 Auswirkungen der Ground-Truth-Korrektur auf den Developmentdatensatz

Zur besseren Nachvollziehbarkeit der Effekte wird in Abbildung 7.9 Einfluss der Korrekturen auf den Ground Truth exemplarisch das im Evaluationssystem implementierte Modal gezeigt. Die Visualisierung verdeutlicht, dass sich bereits vergleichsweise kleine Anpassungen einzelner Werte auf die Gesamtkennzahlen auswirken können.

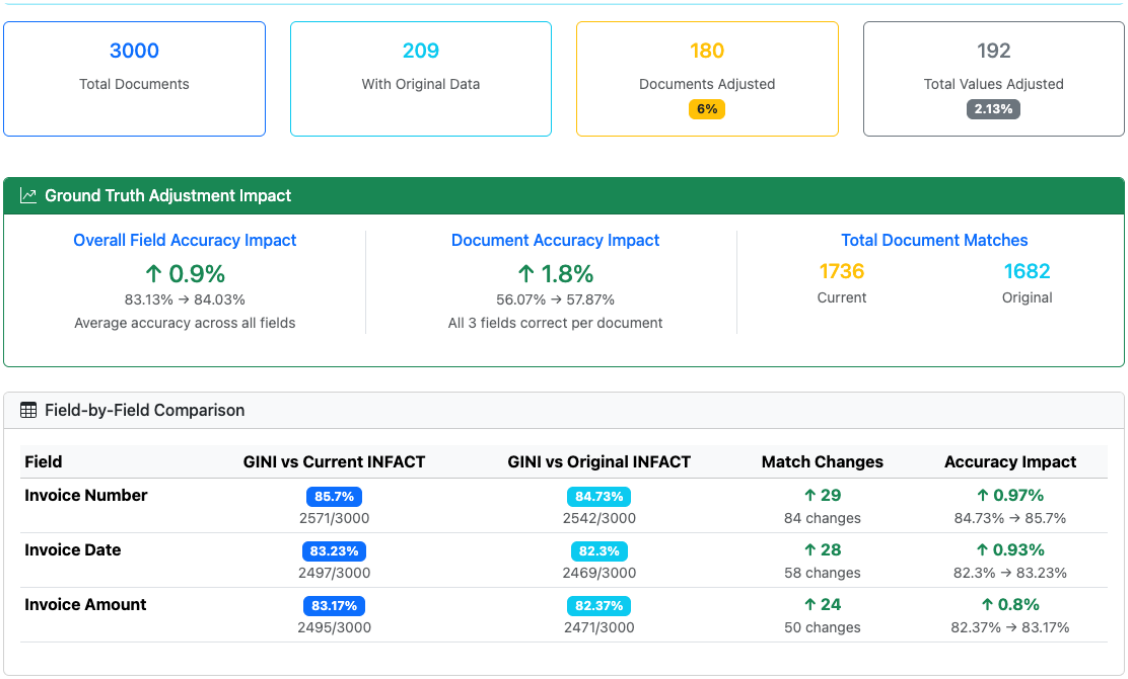


Abbildung 7.9 Einfluss der Korrekturen auf den Ground Truth

7.4 Optimierung der Extraktionsstrategien

Die in Abschnitt 7.2 durchgeführten Tests verdeutlichen, dass Extraktionsfehler nicht allein aus Modellgrenzen resultieren, sondern maßgeblich von Datenqualität, OCR-Genauigkeit und der Struktur der Rechnungen beeinflusst werden.

Ein wiederkehrendes Problem betrifft die fehlerhafte Betragsinterpretation. In manchen Fällen mit Skonto-Angaben wählen die Modelle nicht die eigentliche Gesamtsumme, sondern einen reduzierten Zwischenbetrag (Abbildung 7.10).

Abbildung 7.10 Skontoangaben

Herausfordernd sind auch Abschlagsrechnungen, bei denen mehrere Teilsummen enthalten sind. Hier wählen die Modelle häufig eine Zwischensumme statt des Endbetrags als `invoiceAmount` (Abbildung 7.11).

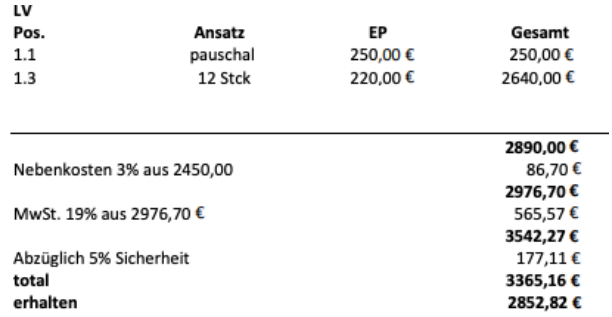


Abbildung 7.11 Zwischensummen

Auch bei Rechnungsnummern treten regelmäßiger Schwierigkeiten auf. Uneinheitliche Bezeichnungen wie „Belegnummer" oder „Beleg" führen dazu, dass Werte aus einem falschen Kontextfeld übernommen werden (Abbildung 7.12).

Abbildung 7.12 verschiedene Nummern

Datumsprobleme stellen eine weitere Fehlerquelle dar. Rechnungen enthalten Datumsangaben in unterschiedlichen Formaten, etwa im US-Format (MM/DD/YYYY, Abbildung 7.12), im EU-Format (DD.MM.YYYY) oder im ISO-Format (YYYY-MM-DD). Zudem sind die Felder nicht immer eindeutig benannt. Manche Dokumente verwenden sowohl „Rechnungsdatum" als auch „Leistungsdatum", was zu Fehlinterpretationen führt. Hinzu kommen Tippfehler, bei denen Datumsangaben inhaltlich fehlerhaft eingetragen sind, etwa „14.20.2023" statt „14.02.2023" (Abbildung 7.14).

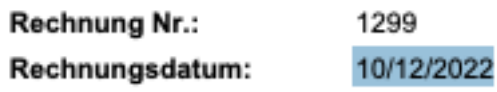


Abbildung 7.14 Amerikanisches Datumsformat

Abbildung 7.14 Typo im Datum

Fehlerhafte Einbettungen stellen eine weitere Quelle für Extraktionsfehler dar. Ein Beispiel zeigt den Unterschied zwischen sichtbarer Darstellung und tatsächlich im PDF hinterlegtem Text. Während das Rechnungsdokument das Datum korrekt als 14.03.2024 ausweist, ist im eingebetteten Text fälschlich 13.09.2023 hinterlegt. Solche Abweichungen entstehen durch fehlerhafte PDF-Generierung und beeinträchtigen die Qualität der Extraktion erheblich (Abbildung 7.15).

Abbildung 7.15 fehlerhafte Einbettung

Neben solchen Fehlern in generierten PDFs treten auch bei OCR-basierten Textlayern Probleme auf. Typisch sind Verwechslungen ähnlich aussehender Zeichen („0" und „O", „1" und „I") oder fehlerhafte Segmentierungen. Hinzu kommen fehlerhafte Text-Embeddings, bei denen semantisch ähnliche Begriffe falsch zugeordnet werden (Abbildung 7.16).

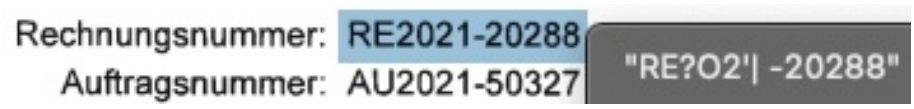


Abbildung 7.16 fehlerhafte OCR-Einbettung

Diese Beispiele zeigen, dass Unschärfen und Mehrdeutigkeiten in Rechnungsdokumenten zu fehlerhaften Ergebnissen führen können, sowohl bei LLMs als auch bei manuellen Bearbeitungen. Wie bereits in Abschnitt 7.3 dargestellt, sind auch menschliche Korrekturen fehleranfällig. Die im Folgenden untersuchten Optimierungsstrategien (Zero-Shot und Few-Shot) basieren auf den in Abschnitt 4.3.3 eingeführten Konzepten und werden unter den in Abschnitt 5.6 beschriebenen Evaluationsbedingungen getestet. Ziel ist es, typische Problemklassen gezielt zu adressieren und die Extraktion insgesamt robuster gegenüber Unsicherheiten zu machen.

7.4.1 Zero-Shot-Optimierung

Die Zero-Shot-Optimierung bildet eine wesentliche Grundlage, um die Rechnungsdatenextraktion ohne Beispielinputs zu verbessern und eine stabile Baseline für nachgelagerte Verfahren festzulegen. Ziel ist es herauszufinden, wie sich verschiedene Prompt-Strategien auf die Ergebnisqualität auswirken und welche Variante den zuverlässigsten Ausgangspunkt für weitere Tests liefert.

Es werden drei Ansätze getestet, die sich an den in eingeführten Prompting-Strategien orientieren. Der erste ist ein restriktiver Prompt mit klaren Format- und Validierungsvorgaben („restriktiv"). Der zweite ist ein bewusst minimalistischer Prompt mit nur den nötigsten Instruktionen („einfach"). Der dritte ist ein strukturierter Chain-of-Thought-Ansatz (CoT), der das Modell schrittweise durch die Extraktion führt.

Die restriktive Variante integriert Instruktionen und Validierungsregeln, um diese Fehlertypen zu adressieren, etwa Verwechslungen von Belegnummern und Rechnungsnummern, uneinheitliche Datumsformate oder die Extraktion von Zwischensummen als Gesamtbetrag (Abbildung 7.17). Der einfache Ansatz testet dagegen stärker das impliziten des Modells. Er reduziert die Anweisungen bewusst auf ein Minimum und prüft, ob die Modelle auch ohne detaillierte Vorgaben korrekte Ergebnisse liefern, solange eine minimale Struktur für die JSON-Ausgabe vorgegeben ist (Abbildung 7.18). Diese Variante ist zugleich relevant für die Nullhypothese H_{02}, nach der zusätzliche Instruktion im Prompt keine signifikanten Verbesserungen der Extraktionsleistung bewirken. Der CoT-Ansatz hingegen führt das Modell explizit schrittweise durch den Extraktionsprozess und soll dadurch die Genauigkeit der Ergebnisse erhöhen (Abbildung 7.19).

```
You extract invoice information from German invoices into JSON with strict type valida-
tion.
Return only raw JSON without formatting or explanations. invoiceAmount is gross/brutto and
always in EUR – also possible: Offene Forderung –
never netto unless explicitly exempt from VAT – prefer Rechnung or Rechnungsnummer, but
Belegnummer is a valid fallback. Dates use ISO format (YYYY-MM-DD). Use null for missing
fields. Strings must be strings, Doubles must be numbers with decimal points, Integers
must be whole numbers. – If you are unconfident, also return the alternative results.
```

Abbildung 7.17 Restriktiver Prompt mit Validierungsregeln

```
Extrahiere Folgende Rechnungsfelder in folgender JSON-Struktur:
{ "invoiceNumber": "String", "invoiceDate": "Date", "invoiceAmount": "Double"}

Hier der Text: $text
```

Abbildung 7.18 einfacher Ansatz

```
You are an expert system for extracting invoice information from text.

Think step by step before giving the final answer:
1. Identify all candidate values for each field.
2. Check each candidate against the required data types and formats:
   - invoiceNumber: Use the exact extracted value without any changes, shortening,
     or normalization. Keep all prefixes (e.g., 'RE-', 'INV-'), hyphens, and
     special characters exactly as they appear in the text.
   - invoiceDate: Convert all dates to ISO format (YYYY-MM-DD). If a date is in
     MM/DD/YYYY (US format), convert it to ISO.
   - invoiceAmount: Numeric (Double), use a dot (.) as the decimal separator,
     no thousands separator. Convert "1.200,50" → "1200.50".
3. Determine the most plausible invoiceAmount: [...]
```

Abbildung 7.19 Chain-of-Thought-Prompt

Die Ergebnisse zeigen deutliche Unterschiede zwischen den Strategien (Abbildung 7.21 Ergebnisübersicht der Zero-Shot-Optimierungsstrategien GPT-4.1 (vollständige Tabelle im Anhang 3.4)

Abbildung 7.22 - Abbildung 7.22). Bei GPT-4.1 liefert der einfache Prompt, der stärker auf implizites Vorwissen, bessere Resultate als die restriktive Variante, was darauf hinweist, dass zu detaillierte Vorgaben das natürliche Reasoning dieses Modells einschränken können. Bei Claude 3 Sonnet liegt der restriktive Ansatz hingegen vor dem einfachen. Bei Gemma 3 27B fallen die Unterschiede insgesamt geringer aus, was darauf hindeutet, dass mittelgroße Modelle stärker von klaren Strukturen profitieren. Gleichzeitig erreichen alle drei Modelle mit dem CoT-An-

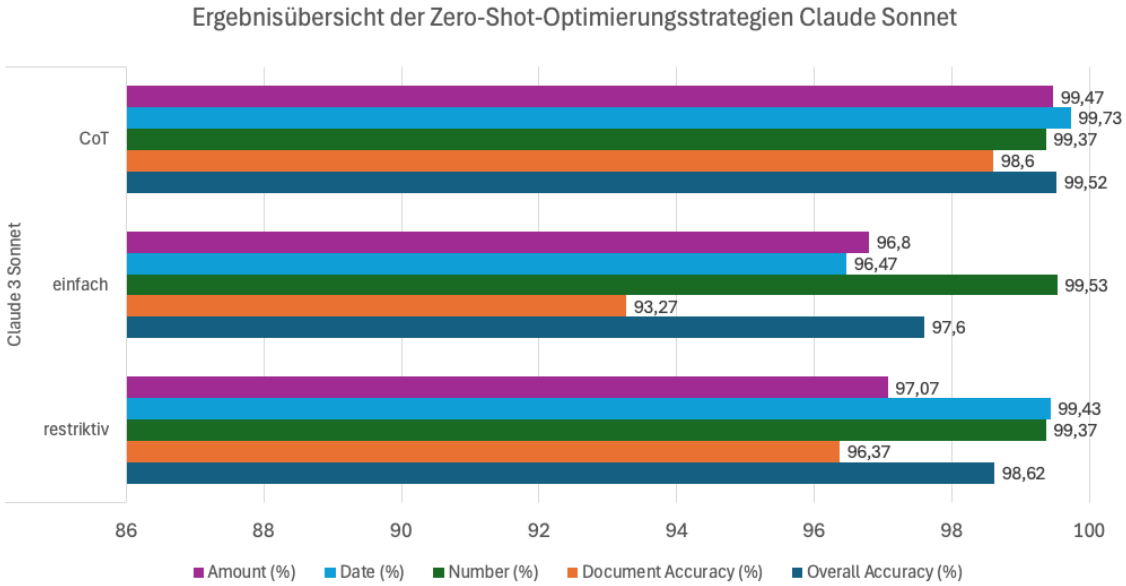
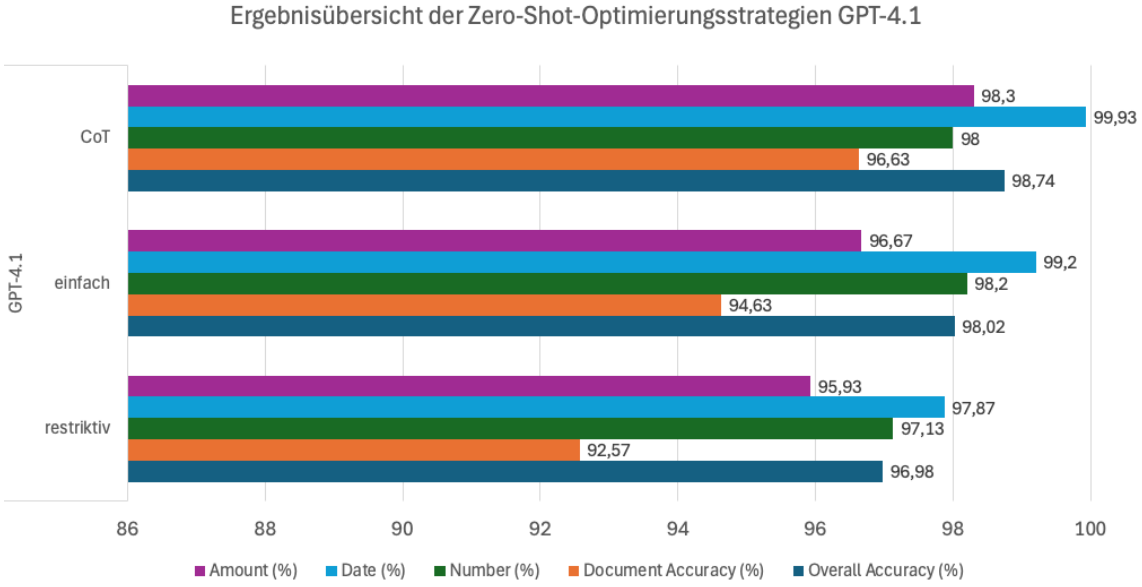


Abbildung 7.20 Ergebnisübersicht der Zero-Shot-Optimierungsstrategien Claude Sonnet (vollständige Tabelle im Anhang 3.4)



satz die besten Ergebnisse. Am stärksten profitiert dabei Claude 3 Sonnet, dass insgesamt die höchsten Genauigkeitswerte erzielt.

Abbildung 7.21 Ergebnisübersicht der Zero-Shot-Optimierungsstrategien GPT-4.1 (vollständige Tabelle im Anhang 3.4)

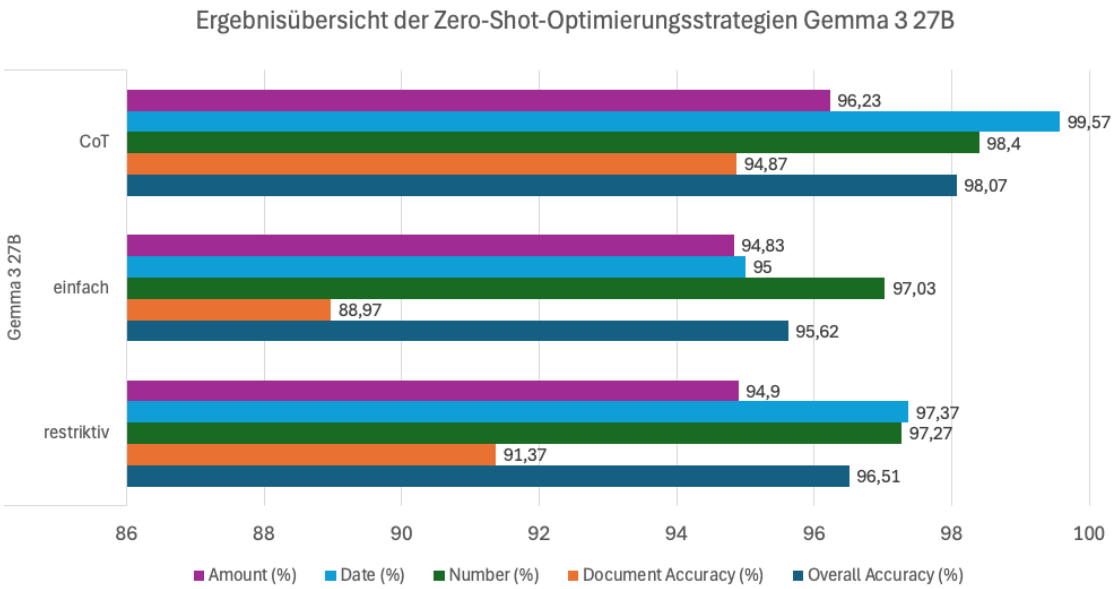


Abbildung 7.22 Ergebnisübersicht der Zero-Shot-Optimierungsstrategien Gemma 3 27B (vollständige Tabelle im Anhang 3.4)

7.4.2 Few-Shot-Optimierung

Die Few-Shot-Optimierung ergänzt die Zero-Shot-Strategien und verfolgt einen alternativen Ansatz zur Verbesserung der Extraktionsqualität. Im Unterschied zu rein instruktionalen Prompts steht hier die Nutzung konkreter Beispiele im Vordergrund, die typische Fehlermuster adressieren. Die Beispiele basieren auf markanten Fehlern aus der Baseline-Evaluierung, sodass gezielt Problemfälle für jedes Extraktionsfeld abgedeckt werden.

Die Strategie wird in zwei Varianten umgesetzt. Der klassische Ansatz verwendet vier synthetisierte Beispiele, die reale Extraktionsfehler nachbilden. Dazu gehören Forderungsabrechnungen mit Eigenanteil, Teilrechnungen mit Anzahlungen sowie Rechnungen mit paralleler Nutzung von Belegnummer und Rechnungsnummer. Ein Auszug aus diesem Few-Shot-Classic-Prompt ist in Abbildung 7.23 dargestellt. Dieser Ansatz verzichtet bewusst auf zusätzliche Reasoning-Schritte und setzt ausschließlich auf die direkte Demonstration der korrekten Extraktionslogik. Der zweite Ansatz kombiniert diese Beispiele mit dem erfolgreichen Chain-of-Thought-Systemprompt (Abbildung 7.19) aus der Zero-Shot-Optimierung, um beide Verfahren zu vereinen.

```
Example 1: Forderungsabrechnung mit Eigenanteil
RECHNUNG Nr.: R24-5923, Datum: 30.12.2024
Summe 166,80 Eigenanteil 26,68 Forderung 140,12
→ {"invoiceNumber": "R24-5923", "invoiceDate": "2024-12-30", "invoiceAmount": 140.12}

Example 4: Teilrechnung mit Anzahlung
TEILRECHNUNG Rechnungsnummer: RE2021-20329, Datum: 23.04.2021
Treppenlift FLOW X 9.159,66 EUR zzgl. 1.740,33 EUR MwSt = Gesamt 10.900,00 EUR
Anzahlung 30% 2.747,90 EUR zzgl. 19% MwSt 522,10 EUR
Anzahlungsbetrag 3.270,00 EUR
Restbetrag 7.630,00 EUR wird später berechnet.
→ {"invoiceNumber": "RE2021-20329", "invoiceDate": "2021-04-23", "invoiceAmount": 3270.0}
```

Abbildung 7.23 Ausschnitt Few Shot Prompt

Die Ergebnisse in Abbildung 7.24 verdeutlichen modellspezifische Unterschiede zwischen den Strategien. Claude 3 Sonnet profitiert am stärksten von der Kombination mit CoT und erzielt die insgesamt höchsten Genauigkeitswerte. Bei GPT-4.1 liegen der klassische und der kombinierte Ansatz nahezu gleichauf, was darauf hindeutet, dass zusätzliche Reasoning-Anweisungen hier keinen klaren Vorteil bringen. Bei Gemma 3 27B zeigt sich ein gegenteiliger Effekt. Der klassische Ansatz liefert leicht bessere Ergebnisse als die kombinierte Variante. Dies unterstreicht, dass der Nutzen kombinierter Strategien stark vom verwendeten Modell abhängt.

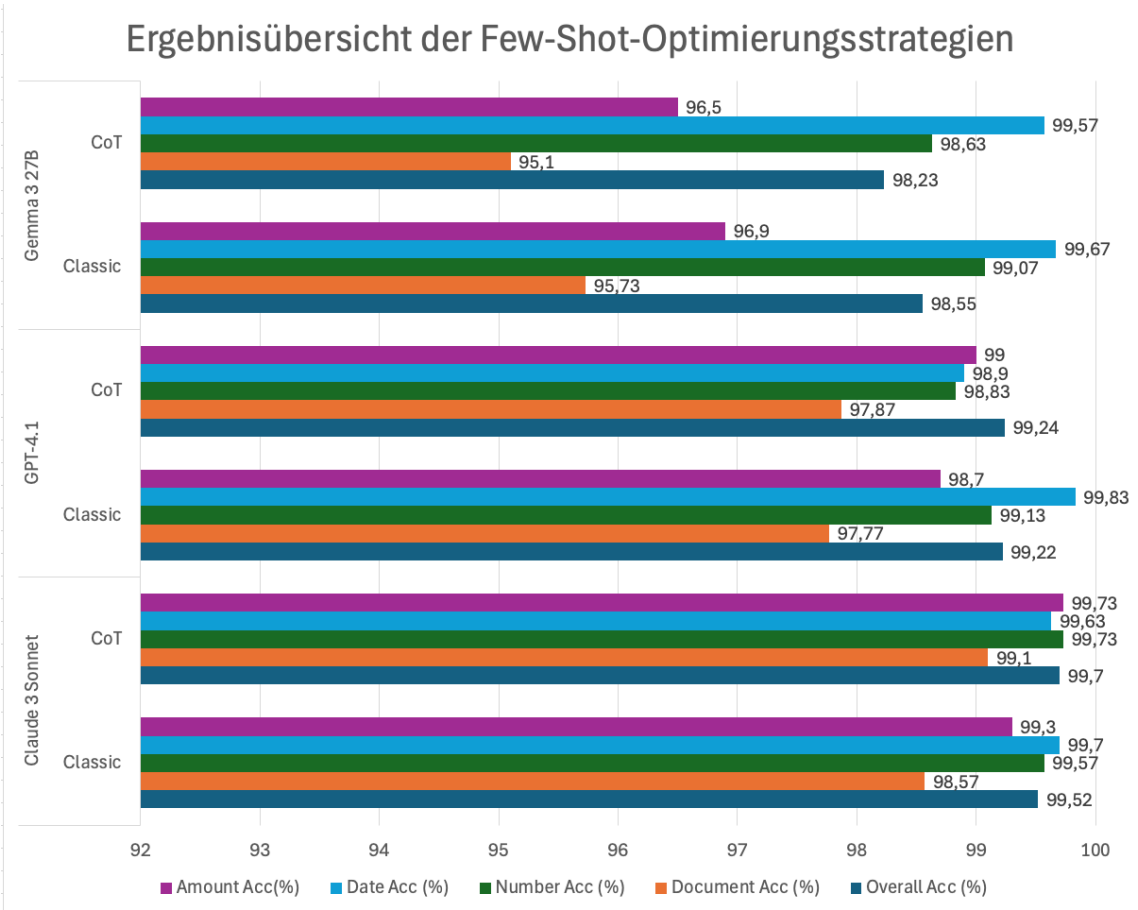


Abbildung 7.24 Ergebnisübersicht der Few-Shot-Optimierungsstrategien (Vollständige Tabelle im Anhang 3.5)

7.5 Evaluationslauf

Die finale Evaluationsphase wird auf dem in Abschnitt 7.1.3 beschriebenen Evaluationsdatensatz durchgeführt, der sich in Zusammensetzung und Komplexität von dem für die Entwicklungs- und Optimierungsphase genutzten Datensatz unterscheidet. Ziel ist es, die in Abschnitt 7.4 definierten Strategien unter identischen, reproduzierbaren Rahmenbedingungen gegeneinander und gegen die bestehende OCR-Lösung zu testen, um deren Leistungsfähigkeit in einem realitätsnahen Szenario belastbar zu bewerten.

Für den Evaluationslauf werden fünf Strategien ausgewählt. Dazu gehören das Zero-Shot-Prompt mit CoT, der klassische Few-Shot-Prompt mit Beispielen typischer Fehlerszenarien, der kombinierte Few-Shot-CoT-Ansatz, das in Abschnitt 7.2.1 definierte Baseline-Prompt sowie der einfache Prompt zur Prüfung der Nullhypothese H_{02}. Diese Strategien repräsentieren die wesentlichen in der Optimierungsphase untersuchten Varianten und decken sowohl minimale Instruktionsszenarien als auch komplexe Reasoning-Ansätze ab.

Alle Evaluationsdurchläufe werden mit den empirisch bestimmten Idealparametern ausgeführt, um die Vergleichbarkeit sicherzustellen. Dabei kommen die folgenden Einstellungen zur Anwendung:

Modell	$temperature$	top_p	max_tokens	$seed$
Claude 3 Sonnet (20250219)	0	1	200	–
GPT-4.1	0	1	200	3333
Gemma 3 27B-IT	1	1	200	3333

Tabelle 7.10 Parameterkonfigurationen für den Evaluationslauf

Wie in Abschnitt 7.3 erläutert, weisen die Infact-Referenzdaten systematische Abweichungen auf, die andernfalls zu verzerrten Ergebnissen führen würden. Deshalb werden im Rahmen der Evaluationsphase auch die Referenzdaten des Evaluationsdatensatzes manuell nachkorrigiert. Auf diese Weise wird eine faire und konsistente Vergleichsbasis geschaffen. Die resultierenden Veränderungen in den Metriken sind in Tabelle 7.11 dargestellt.

Feld	**vor Korrektur**	**nach Korrektur**	**Veränderung**
$Number\ Acc$	96,92%	96,72%	-0,20%
$Date\ Acc$	96,30%	96,51%	-0,11%
$Amount\ Acc$	82,37%	83,17%	+0,21%
$Overall\ Acc$	83,13%	84,03%	-0,04%
$Document\ Acc$	56,07%	57,87%	-0,14%

Tabelle 7.11 Auswirkungen der Ground-Truth-Korrektur auf den Evaluationsdatensatz

Durch die Festlegung einheitlicher Parameter (Tabelle 7.10) und die konsequente manuelle Nachprüfung potenziell fehlerhafter Ergebnisse wird sichergestellt, dass alle Strategien unter exakt vergleichbaren Bedingungen getestet werden können. So lassen sich Unterschiede in den Resultaten eindeutig auf Modell- und Prompting-Strategie-Effekte zurückführen, ohne dass Verzerrungen durch zufällige Sampling-Effekte, fehlerhafte Ground-Truth-Daten oder variierende Textextraktionen entstehen. Diese Vorgehensweise bildet die methodische Grundlage für die im folgenden Kapitel dargestellte Ergebnisanalyse.

8 Ergebnisse

Die finale Evaluation erfolgt auf Basis des in Abschnitt 7.5 beschriebenen Setups und überprüft die Leistungsfähigkeit der entwickelten Strategien auf dem unabhängigen Evaluationsdatensatz. Im Mittelpunkt steht die Gesamtleistung der Modelle, ergänzt durch eine vertiefte Betrachtung typischer Fehlermuster und ihrer Ursachen. Darüber hinaus werden die aufgestellten Hypothesen überprüft, die Resultate mit der bisherigen OCR-Lösung verglichen und Laufzeit- sowie Tokenaspekte ausgewertet. Abschließend werden die Grenzen des rein textbasierten Ansatzes diskutiert.

8.1 Gesamtergebnisse

Die Ergebnisse der finalen Evaluation sind in Abbildung 8.1 dargestellt. Der Überblick verdeutlicht, dass sich die LLM-basierten Ansätze in allen Strategien klar von der bisherigen OCR-Lösung absetzen. Während Gini mit einer $Document\ Accuracy$ von 87,24% deutlich zurückliegt, erreichen selbst die schwächeren Strategien der LLMs noch Werte oberhalb von 94%. Besonders deutlich wird, dass die optimierten Few-Shot- und Few-Shot-CoT-Strategien bei allen drei Modellen die höchsten Werte erzielen. Claude 3 Sonnet liefert hier mit 99,36% $Document\ Accuracy$ nahezu fehlerfreie Extraktionen, dicht gefolgt von

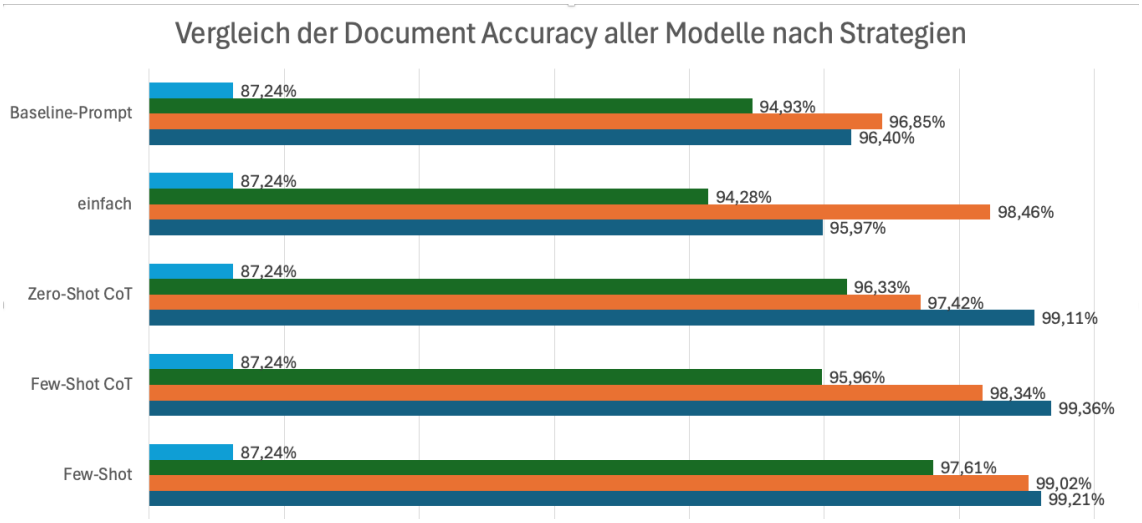


Abbildung 8.1 Vergleich der $Document\ Accuracy$ aller Modelle nach Strategien (vollständige Tabelle im Anhang 3.6)

GPT-4.1 mit 99,02% im Few-Shot-Setup. Gemma 3 27B-IT bleibt mit 97,61% etwas darunter, liegt aber ebenfalls klar über der Baseline.

Die Abbildung macht zudem deutlich, dass die Strategien mit Beispiel-basierten Prompts durchweg überlegen sind, während einfache oder Baseline-Prompts nur im Mittelfeld liegen. Auffällig ist, dass GPT-4.1 in der einfachen Variante überraschend stark abschneidet und kaum hinter die optimierten Ansätze zurückfällt, während Claude 3 Sonnet und Gemma 3 27B deutlicher von zusätzlichen Beispielen und Reasoning-Schritten profitieren.

Eine genauere Betrachtung der feldspezifischen Ergebnisse findet sich in Abbildung 8.2. Hier wird sichtbar, dass die Spitzenwerte nicht auf einzelne Felder beschränkt sind, sondern über alle Felder hinweg konsistent auftreten. Claude 3 Sonnet erreicht in allen optimierten Strategien über 99% $Document\ Accuracy$ und weist mit lediglich 0,25 Prozentpunkten zwischen der besten und der schlechtesten Variante eine sehr geringe Spannweite auf. GPT-4.1 bewegt sich in der Few-Shot-Variante auf einem ähnlich hohen Niveau, während Gemma 3 27B mit 97,61% $Document\ Accuracy$ zwar sehr gute, aber im Vergleich etwas niedrigere Werte erzielt.

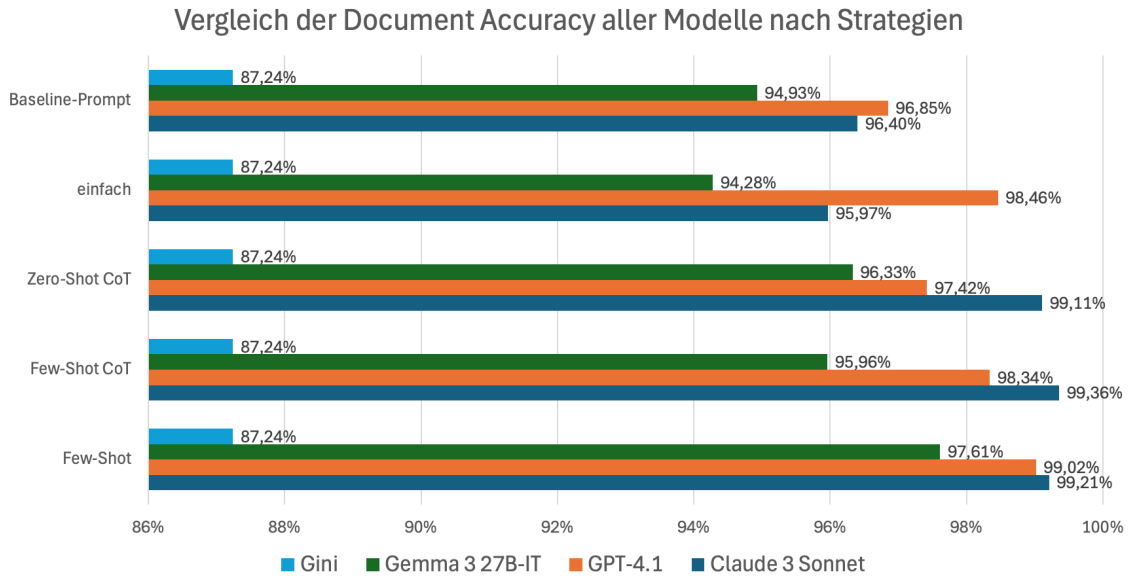


Abbildung 8.2 Detailanalyse der Evaluationsergebnisse der besten Strategie je Model (vollständige Tabelle im Anhang 3.6)

Im direkten Vergleich mit der OCR-Lösung wird der Vorteil der LLM-Ansätze besonders deutlich. Während Gini gerade bei Datums- und Betragsfeldern wiederholt Fehler produziert, überschreiten die LLMs hier durchweg die 98-Prozent-Marke. Die Detailanalyse bestätigt damit, dass die Überlegenheit der LLM-basierten Verfahren nicht nur in den aggregierten Kennzahlen sichtbar ist, sondern auch in sämtlichen Einzelfeldern konsistent nachgewiesen werden kann.

8.2 Fehlermuster der eingebetteten Texte

Die Analyse der Fehlfälle zeigt, dass die Hauptursache für verbleibende Fehler nicht in den Modellen selbst liegt, sondern in der Qualität der im PDF eingebetteten Texte (Textlayer). Dieser Textlayer wird nicht von der Evaluationsplattform selbst erzeugt, sondern stammt von Systemen Dritter, die entweder nativ Text in die PDF-Datei einbetten oder eine vorgeschaltete OCR-Erkennung auf bildbasierten PDFs durchführen und deren Ergebnisse einbetten. In beiden Fällen ist die Extraktion vollständig von der Korrektheit und Konsistenz dieser extern bereitgestellten Zeicheninformationen abhängig.

Bei der besten getesteten Strategie (Claude 3 Sonnet Few-Shot CoT) konnten 478 von 508 Kreditoren, also unterschiedliche Layouts, vollständig fehlerfrei verarbeitet werden. Tabelle 8.1 zeigt die Verteilung der Fehlerraten über alle Strategien, einschließlich des bisherigen Gini-Systems. Es wird sichtbar, dass die LLM-basierten Ansätze (Claude, GPT, Gemma) insgesamt eine deutlich höhere Genauigkeit erreichen, während Gini bei einem größeren Anteil der Kreditoren fehlerhafte Extraktionen liefert. Die verbleibenden Fehler lassen sich überwiegend auf inkonsistente OCR- oder Textlayer-Einbettungen zurückführen.

Strategie\Fehler je Kreditor	**= 0%**	**> 0% < 70%**	**> 70%**
Claude Few-Shot CoT	478	19	11
GPT Few-Shot	472	27	9
Gemma Few-Shot	460	37	11
Gini	350	95	63

Tabelle 8.1 Anteil fehlerhafter Dokumente je Kreditoren und Strategie

Die Analyse der Textlayer verdeutlicht verschiedene Fehlermuster, die sich in folgende Kategorien gliedern lassen:

Fehlerhafte oder abweichende Werte im Textlayer

In einigen Fällen werden ähnliche Zeichen fehlerhaft erkannt und falsch eingebettet. Häufig betrifft dies die Verwechslung von Ziffern und Buchstaben wie „0" und „O" oder „1" und „I". Solche Fehler können auch bei Datumsangaben auftreten, wenn Zeichen ausgelassen oder mit anderen Fragmenten vermischt werden (Abbildung 8.3).

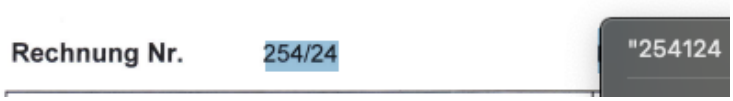


Abbildung 8.3 sichtbares „254/24" versus eingebettetes „254124

Teilweise unterscheidet sich der eingebettete Wert deutlich vom tatsächlich sichtbaren Text. Ein Beispiel ist die fehlerhafte Zusammenführung von Ziffern durch fehlende Trennzeichen (Abbildung 8.4). Solche Abweichungen verändern die Bedeutung der extrahierten Information und können insbesondere bei Beträgen oder Nummernfeldern zu Fehlklassifikationen führen.

Abbildung 8.4 sichtbares Datum „30.07.2024" versus eingebettetes Fragment „…m 26. Juli 20…

Phantomtexte

In einigen Fällen enthält der Textlayer Werte, die im Dokument visuell nicht vorhanden sind (Abbildung 8.5). Diese konkurrieren mit den tatsächlich sichtbaren Zahlen und erschweren die korrekte Auswahl durch das Modell, da konkurrierende Kandidatenwerte entstehen, die nicht plausibel verifizierbar sind.

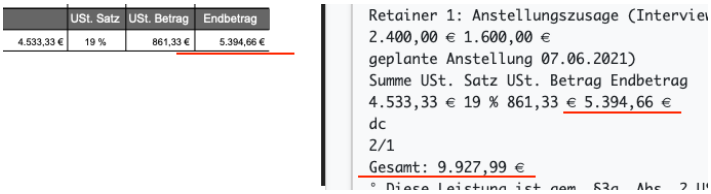


Abbildung 8.5 Phantomtexte

Fehlende semantische Marker

Auch bei identischen Layouts treten unterschiedliche Extraktionsergebnisse auf. In Abbildung 8.9 und Abbildung 8.7 wird die Kundennummer als Rechnungsnummer interpretiert, während in Abbildung 8.8 und Abbildung 8.6 die Belegnummer erkannt wird. Auffällig ist, dass in beiden Fällen die eingebetteten Texte keine Feldbezeichnungen enthalten, wodurch die Modelle auf schwächere Kontextsignale angewiesen sind.

Abbildung 8.9 Sichtbarer Rechnungsblock mit vollständigen Feldbezeichnungen (Rechnung A)

Abbildung 8.8 Sichtbarer Rechnungsblock mit identischem Layout (Rechnung B)

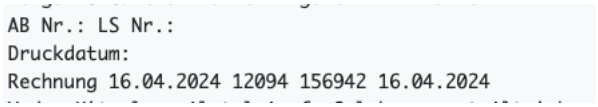


Abbildung 8.7 Eingebetteter Textlayer ohne Feldbezeichnungen zu Abbildung 8.9

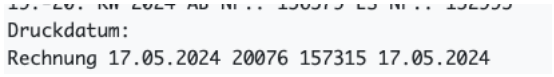


Abbildung 8.6 Eingebetteter Textlayer ohne Feldbezeichnungen zu Abbildung 8.8

Diese Ergebnisse verdeutlichen, dass große Sprachmodelle eine leistungsfähige Möglichkeit für das Postprocessing von Rechnungsdaten darstellen. Gleichzeitig zeigen sie, wie stark die erzielte Genauigkeit von der Qualität des Preprocessings abhängt und damit auch von der Verlässlichkeit der eingebetteten Texte. Die ausschließliche Nutzung solcher Textlayer bietet bei korrekt erzeugten oder nativ generierten Dokumenten eine sehr hohe Extraktionsqualität, bringt jedoch auch Risiken mit sich. Fehlerhafte OCR-Ergebnisse, fragmentierte Einbettungen oder sogar absichtlich manipulierte Textlayer können nicht zuverlässig erkannt oder validiert werden. Dadurch entsteht eine Abhängigkeit von Drittanbietern, deren Textbereitstellung nicht immer überprüfbar ist.

8.3 Evaluation der Hypothesen und Vergleich zur Baseline

Die finale Evaluation der getesteten Modellstrategien erfolgt auf einem unabhängigen Testdatensatz mit 508 verschiedenen Kreditoren und erheblicher Layout-Varianz. Bewertet wird die Extraktionsqualität anhand der vollständigen Korrektheit der drei Felder invoiceNumber, invoiceDate und invoiceAmount. Die Ergebnisse dienen zur Überprüfung der in Abschnitt 2.3 formulierten Hypothesenpaare H_{01} bis H_{03}. Neben der Auswertung beschreibender Kennzahlen wird die statistische Signifikanz mittels dokumentbasierter McNemar-Tests nach Abschnitt 5.3.4 überprüft.

8.3.1 H_{01} Layout-Robustheit

Die getesteten LLM-Strategien erzielen bei der im Evaluationsdatensatz vorhandenen Layout-Varianz, bedingt durch 508 verschiedene Kreditoren, signifikant bessere Ergebnisse als die bestehende OCR-basierte Lösung der Gini GmbH. Während Gini nur eine $Document\ Accuracy$ von rund 87% erreicht, liegen die LLM-basierten Verfahren deutlich höher. Abbildung 8.10 Ergebnisse des McNemar-Tests für Claude 3 Sonnet (Few-Shot-CoT) vs. Gini (n = 10.000) illustriert am Beispiel von Claude 3 Sonnet, dass die Leistungsunterschiede nicht zufällig sind. In 1.247 Fällen liefert Claude korrekte Ergebnisse, bei denen Gini fehlschlägt, während die umgekehrte Konstellation nur 35 Mal auftritt.

Der McNemar-Test weist für diesen Vergleich ein Odds Ratio von 35,6 mit einem 95%-Konfidenzintervall von 25,5 bis 49,9 aus. Dies verdeutlicht eine mehr als 35-fache Überlegenheit von Claude gegenüber Gini bei der Extraktion.

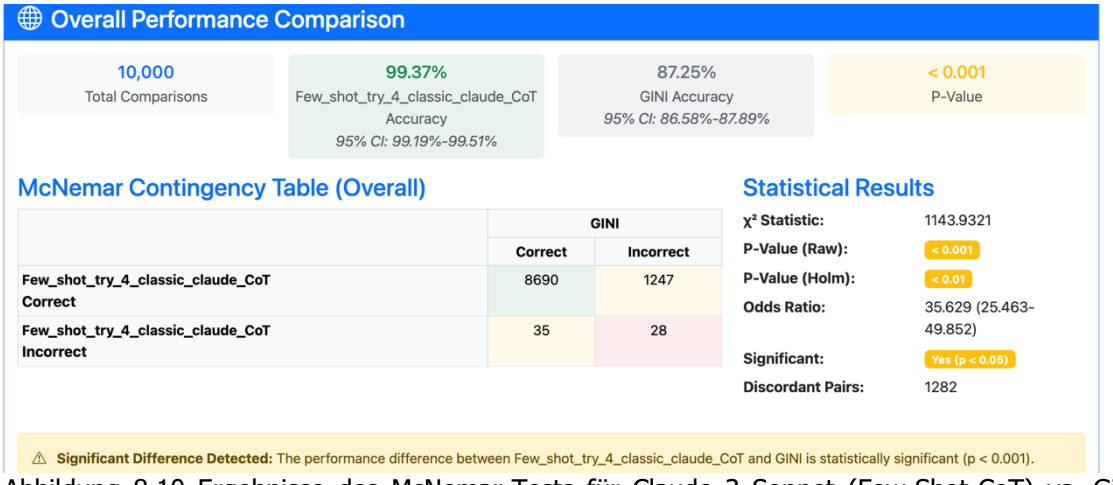


Abbildung 8.10 Ergebnisse des McNemar-Tests für Claude 3 Sonnet (Few-Shot-CoT) vs. Gini (n = 10.000)

Auch die übrigen Strategien zeigen in der statistischen Auswertung klare Vorteile. Tabelle 8.2 fasst die zentralen Ergebnisse zusammen. GPT-4.1 Few-Shot liegt auf einem ähnlich hohen Niveau wie Claude, während Gemma 3 27B mit Few-Shot zwar ebenfalls signifikant besser als Gini abschneidet, im Vergleich aber etwas schwächer bleibt. Die McNemar-Tests ergeben in allen Fällen hochsignifikante Unterschiede, die auch nach Holm-Korrektur bestehen bleiben. Damit kann die Nullhypothese H_{01}, wonach LLMs bei hoher Layout-Varianz keine signifikant besseren Ergebnisse als klassische OCR liefern, zurückgewiesen werden. Stattdessen bestätigt sich die Alternativhypothese H_{A1}, dass LLMs bei starker Layout-Diversität signifikant bessere Ergebnisse erzielen. Für die Gini-Baseline sind keine OR-, χ^2- oder p-Werte angegeben, da sie als Referenzwert dient.

Strategie	$Overall\ Accuracy$ **(%)**	$Document\ Accuracy$ **(%) (95%-CI)**	**McNemar OR (95%-CI)**	**χ^2-Wert**	**p-Wert (Holm-adj.)**
Gini (Baseline)	95,53	87,24 (86,57 – 87,88)	–	–	–
Claude Few-Shot CoT	99,76	99,36 (99,18 – 99,5)	35,63 (25,46 – 49,85)	1143,93	< 0,01
GPT Few-Shot	99,64	99,02 (98,81 – 99,2)	20,0 (15,5 – 25,81)	1064,00	< 0,01
Gemma 3 27B Few-Shot	99,13	97,61 (97,29 – 97,89)	9,56 (7,93 – 11,53)	838,20	< 0,01

Tabelle 8.2 Gesamtergebnisse aller Strategien im Vergleich zur Gini-Baseline

8.3.2 H_{02} Optimierung durch Prompt-Engineering

Die getesteten Promptstrategien zeigen im Vergleich zu einfachen Prompts ein uneinheitliches Bild. Während Claude 3 Sonnet und Gemma 3 27B von der zusätzlichen Strukturierung klar profitieren, fällt das Ergebnis bei GPT-4.1 differenzierter aus. Abbildung 8.11 verdeutlicht dies anhand der $Document\ Accuracy$ im direkten Vergleich zwischen dem einfachen Prompt und jeweils der besten sowie der schlechtesten optimierten Strategie. Claude 3 Sonnet erreicht mit Few-Shot-CoT fast vier Prozentpunkte mehr als der Simple Prompt, auch Zero-Shot-CoT führt zu einer deutlichen Verbesserung. Bei Gemma 3 27B sind die Unterschiede kleiner, aber dennoch durchgehend signifikant. GPT-4.1 hingegen erzielt bereits mit dem Simple Prompt ein sehr hohes Leistungsniveau, das durch Few-Shot leicht gesteigert wird, während Zero-Shot-CoT eine signifikante Verschlechterung führt, wie Tabelle 8.3 anhand eines Odds Ratios von 0,33 zeigt.

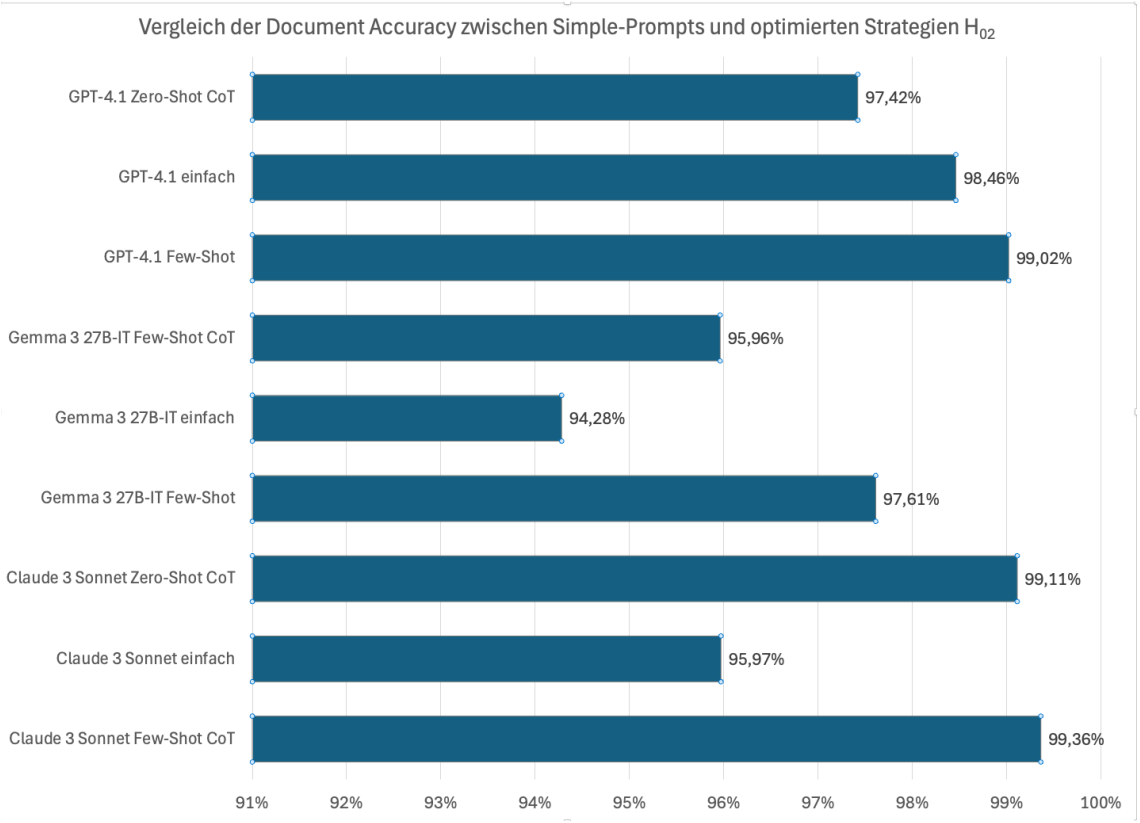


Abbildung 8.11 Vergleich der *Document Accuracy* zwischen Simple-Prompts und optimierten Strategien je Modell

Die McNemar-Tests machen die Unterschiede klar messbar. Bei Claude 3 Sonnet treten strukturierte Strategien in mehr als dreißigmal so vielen Fällen korrekt auf wie die einfache Variante umgekehrt. Bei Gemma 3 27B zeigt Few-Shot eine signifikante Verbesserung im Bereich des Vier- bis Fünffachen gegenüber dem Simple Prompt. Die Few-Shot-Strategie ist rund neunmal häufiger erfolgreich als der Simple Prompt, Zero-Shot-CoT hingegen bleibt deutlich unterlegen.

Tabelle 8.3 fasst die Ergebnisse zusammen und, dass die Hypothese H02 nicht für alle Modelle einheitlich bewertet werden kann. Claude 3 Sonnet und Gemma 3 27B die Hypothese klar widerlegen, gilt sie für GPT-4.1 zumindest teilweise. Die Ergebnisse verdeutlichen, dass die Wirkung von Prompt-Engineering stark vom verwendeten Modell abhängt. Eine allgemeine Aussage über alle Systeme hinweg ist daher nicht möglich, vielmehr sollte die Wahl einer Strategie stets im Kontext des eingesetzten Modells getroffen werden.

Modell	**Strategie**	$\boldsymbol{Document\ Accuracy}$ **(%) (95%-CI)**	**McNemar OR (95%-CI)**	**χ^2-Wert**	**p-Wert (Holm-adj.)**
Claude 3 Sonnet	Simple	95,97 (95,57 – 96,34)	–	–	–
	Few-Shot-CoT	99,36 (99,18 – 99,5)	31,82 (17,46 – 57,99)	318,34	< 0,01
	Zero-Shot-CoT	99,11 (98,91 – 99,28)	15,27 (9,92 – 23,51)	375,41	< 0,01
GPT-4.1	Simple	98,46 (98,2 – 98,68)	–	–	–
	Few-Shot	99,02 (98,81 – 99,2)	9 (4,12 – 19,65)	44,8	< 0,01
	Zero-Shot-CoT	97,42 (97,09 – 97,71)	0,33 (0,24 – 0,46)	52	< 0,01
Gemma 3 27B	Simple	94,28 (93,81 – 94,72)	–	–	–
	Few-Shot	97,61 (97,29 – 97,89)	4,36 (3,51 – 5,43)	208,83	< 0,01
	Few-Shot-CoT	95,96 (95,56 – 96,33)	1,61 (1,38 – 1,87)	39,20	< 0,01

Tabelle 8.3 Vergleich zwischen einfacher („Simple") und alternativer Strategie pro Modell ($Document\ Accuracy$ und McNemar-Test) (n = 10.000)

8.3.3 H_{03} Generalisierbarkeit auf unabhängige Daten

Die Generalisierbarkeit der getesteten Strategien wird überprüft, indem die Ergebnisse auf einem unabhängigen Evaluationsdatensatz den Resultaten aus der Optimierungsphase gegenübergestellt werden. Hypothese H_{03} nimmt an, dass LLMs auf unabhängigen Daten eine signifikant schlechtere Leistung zeigen als bei den in der Optimierungsphase genutzten Entwicklungsdaten. Der Evaluationsdatensatz umfasst dabei eine eigenständige Stichprobe von Rechnungen mit vergleichbarer Layout-Diversität, die nicht in die Optimierung eingeflossen ist (Abbildung 8.4).

Die Analyse konzentriert sich auf die jeweils besten Strategien der getesteten Modelle. Im Vergleich zeigt sich ein stabiles Leistungsbild ohne signifikante Einbußen. Claude 3 Sonnet und GPT-4.1 erreichen nahezu identische Ergebnisse in beiden Datensätzen, während Gemma 3 27B im Evaluationsdatensatz sogar eine leichte Verbesserung erzielt. Alle Abweichungen bewegen sich innerhalb der erwartbaren Schwankungsbreite der Wilson-Konfidenzintervalle und sind statistisch nicht signifikant. Dieses Ergebnis lässt sich durch die unterschiedliche Zusammensetzung der Datensätze erklären. Während der Developmentdatensatz gezielt komplexere Fälle enthielt, um Optimierungspotenziale sichtbar zu machen, bildet der Evaluationsdatensatz eine breiter angelegte und repräsentative Stichprobe ab. Ein direkter Leistungsvergleich ist daher nur eingeschränkt sinnvoll, da tendenziell höhere Genauigkeiten zu erwarten sind. Entscheidend ist, dass die Strategien trotz fehlender Optimierung stabil und teilweise sogar besser abschneiden. Dies belegt eine robuste Generalisierbarkeit und zeigt keine Hinweise auf Overfitting.

Modell / Strategie	**Developmentdatensatz % (95%-CI)**	**Evaluationsdatensatz % (95%-CI)**
Claude 3 Sonnet (Few-Shot-CoT)	99,20 (98,81 – 99,46)	99,36 (99,18 – 99,50)
GPT-4.1 (Few-Shot)	99,00 (98,58 – 99,30)	99,02 (98,81 – 99,20)
Gemma 3 27B (Few-Shot)	95,73 (94,95 – 96,40)	97,61 (97,29 – 97,89)

Tabelle 8.4 Vergleich der Modellleistung auf Developmentdatensatz und Evaluationsdatensatz ($Document\ Accuracy$)

Die Ergebnisse widersprechen der Annahme, dass LLMs auf unabhängigen Daten systematisch schlechtere Leistungen erzielen. Stattdessen bestätigen sie eine konsistente Extraktionsqualität über verschiedene Dokumentmengen hinweg. Die Nullhypothese H_{03} wird zurückgewiesen und die Alternativhypothese H_{A3} bestätigt.

8.4 Token und Laufzeit

Neben der reinen Extraktionsqualität wurde auch die technische Performanz der Strategien in Form von Tokenverbrauch und Laufzeiten untersucht. Diese Kennzahlen sind nicht primärer Bestandteil der Evaluation, verdeutlichen jedoch praxisrelevante Unterschiede zwischen den Modellen.

Die in Tabelle 8.5 dargestellten Ergebnisse beziehen sich auf die im Rahmen der Untersuchung eingesetzten Strategien, die ausschließlich die Felder Rechnungsnummer, Datum und Betrag extrahieren. Die Werte zeigen, dass Strategien mit komplexeren Prompts im Durchschnitt mehr Tokens pro Anfrage benötigen und damit auch längere Antwortzeiten aufweisen. So liegt die durchschnittliche Verarbeitungsdauer bei Claude 3 Sonnet mit Few-Shot-CoT bei rund 1,4 Sekunden, während GPT-4.1 mit vergleichbarer Strategie unter einer Sekunde bleibt. Bei Gemma 3 27B bewegen sich die Laufzeiten mit 1,6 bis 1,8 Sekunden auf einem höheren Niveau, was sowohl auf die Modellgröße als auch auf die promptbedingte Tokenzahl zurückzuführen ist.

Modell / Strategie	**Avg Tokens (± σ)**	**Avg Prompt Tokens (± σ)**	**Avg Completion Tokens (± σ)**	**Avg Dauer ms (± σ)**
Claude 3 Sonnet CoT	1454 ± 533	1410 ± 533	45 ± 2	1459 ± 528
Claude 3 Sonnet Few-Shot-CoT	2083 ± 533	2046 ± 533	37 ± 2	1387 ± 349
GPT-4.1 CoT	1132 ± 405	1097 ± 405	36 ± 2	951 ± 746
GPT-4.1 Few-Shot	1368 ± 406	1338 ± 405	31 ± 2	975 ± 672
Gemma 3 27B CoT	1341 ± 531	1287 ± 530	55 ± 4	1783 ± 373
Gemma 3 27B Few-Shot-CoT	2056 ± 531	2015 ± 530	41 ± 2	1638 ± 391

Tabelle 8.5 Token- und Laufzeitstatistiken der untersuchten Strategien

Darüber hinaus wurden praxisnähere „Naive"-Varianten getestet, die nicht nur drei Felder, sondern ein vollständiges JSON mit mehreren Rechnungsfeldern extrahieren. Diese Varianten wurden ergänzend einbezogen, um die Übertragbarkeit in realistischeren Anwendungsszenarien zu prüfen. Der breitere Extraktionsumfang führt erwartungsgemäß zu höheren Completion-Tokens und deutlich längeren Laufzeiten. So bewegen sich die durchschnittlichen Antwortzeiten hier zwischen etwa 6 und 12 Sekunden, mit Mittelwerten von rund 9 bis 10 Sekunden. Für automatisierte Back-End-Prozesse mag dies noch akzeptabel sein, im Kontext direkter Nutzerinteraktionen, etwa über einen Call-to-Action in einem B2C-Frontend, kann eine solche Verzögerung jedoch als problematisch empfunden werden. Bereits Wartezeiten über 1 Sekunde unterbrechen den Arbeitsfluss, und bei mehr als 10 Sekunden geht die Nutzeraufmerksamkeit deutlich zurück [95].

Modell / Strategie	**Avg Tokens (± σ)**	**Avg Prompt Tokens (± σ)**	**Avg Completion Tokens (± σ)**	**Avg Dauer ms (± σ)**
GPT-4.1 Naive	1748 ± 714	1160 ± 468	589 ± 307	5862 ± 4514
Claude 3 Sonnet Naive	2157 ± 846	1512 ± 655	646 ± 278	9270 ± 3635
Gemma 3 27B Naive	1780 ± 773	1271 ± 530	510 ± 298	11961 ± 7583

Tabelle 8.6 Token- und Laufzeitstatistiken der praxisnäheren Naive-Strategien

8.5 Grenzen des rein textbasierten Ansatzes

Die durchgeführten Experimente zeigen, dass LLMs bei Dokumenten mit eingebetteten Textlayern eine sehr hohe Extraktionsgenauigkeit erreichen. Die Leistungsfähigkeit hängt jedoch vollständig von der Qualität dieser Einbettungen ab. Im Rahmen des Versuchsaufbaus wurden daher ausschließlich Dokumente mit vorhandenem Textlayer untersucht, während reine Bild-PDFs ohne eingebetteten Text sowie Dateien mit problematischen Codierungen (z. B. CID oder Unicode-Ersatzzeichen, vgl. Abschnitt 7.1.2) vorab ausgeschlossen wurden. Die Ergebnisse beziehen sich somit unmittelbar auf Szenarien, in denen ein Textlayer vorhanden ist, unabhängig davon, ob dieser aus der Originalquelle oder durch vorgelagerte OCR-Verfahren erzeugt wurde.

Diese Abhängigkeit wurde insbesondere in den Fehlermustern in Abschnitt 8.2 sichtbar, wo fehlerhafte oder inkonsistente Einbettungen die Modellleistung direkt

beeinträchtigten. Fehlerhafte oder manipulierte Textlayer können von den Modellen in einem rein textbasierten Ansatz nicht zuverlässig erkannt oder validiert werden, wodurch die Robustheit des Ansatzes begrenzt bleibt. Neben unbeabsichtigten Fehlern eröffnet dies auch ein mögliches Angriffsszenario, da gezielt veränderte Einbettungen die Extraktionsergebnisse verfälschen können

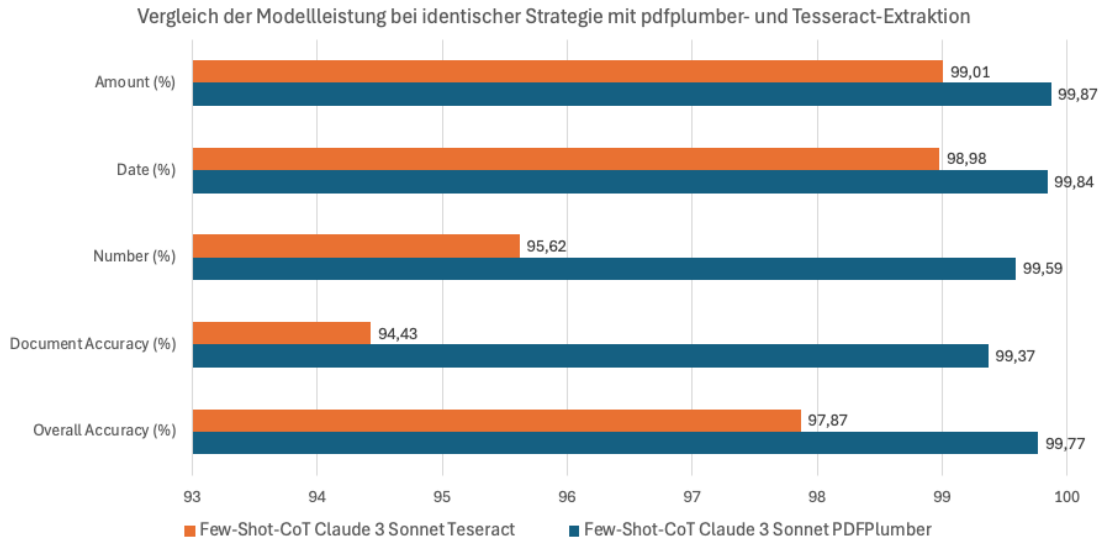


Abbildung 8.12 Vergleich der Modellleistung bei identischer Strategie mit pdfplumber- und Tesseract-Extraktion (vollständige Tabelle im Anhang 3.7)

Die Ergebnisse zeigen den Einfluss der Extraktionsmethode auf die Genauigkeit bei identischer Strategie (Claude 3 Sonnet Few-Shot CoT). Während die Textlayer-basierte Extraktion mit pdfplumber in allen Metriken nahezu perfekte Ergebnisse erzielt ($Overall\ Accuracy$ 99,77%), liegen die Werte bei der OCR-basierten Extraktion mit Tesseract deutlich niedriger ($Overall\ Accuracy$ 97,87%). Beide Verfahren wurden jedoch in einer einfachen Implementierung verwendet und repräsentieren damit nicht zwingend die bestmögliche Leistungsfähigkeit der Verfahren. Der Vergleich verdeutlicht jedoch die enge Kopplung der LLM-Leistung an die Qualität des zugrunde liegenden Textlayers.

Für ein produktives Setup empfiehlt es sich daher, die LLM-Auswertung durch ein zusätzliches optisches Verfahren abzusichern, um die Plausibilität der eingebetteten Inhalte zu prüfen und fehlerhafte Texte frühzeitig zu erkennen. Langfristig könnten multimodale Modelle, die direkt auf visuelle Eingaben zugreifen, diese Lücke schließen und die Abhängigkeit von Drittanbieter-Textlayern weiter reduzieren. Die Ergebnisse verdeutlichen somit nicht nur die aktuellen Grenzen rein textbasierter Verfahren, sondern liefern auch einen klaren Ausblick auf notwendige Weiterentwicklungen.

9 Diskussion und Handlungsempfehlungen

Die in dieser Arbeit erzielten Ergebnisse bilden eine fundierte Grundlage für die Beantwortung der eingangs formulierten Forschungsfragen und deren Einordnung in den praktischen Kontext der aifinyo AG. Ziel dieses Kapitels ist es, die zentralen Schlussfolgerungen aus den Experimenten zu verdichten, ihre Relevanz für den produktiven Einsatz einzuordnen und den Geltungsbereich der Ergebnisse transparent abzugrenzen. Zunächst werden die Forschungsfragen anhand der finalen Evaluationsergebnisse beantwortet. Darauf folgt eine Reflexion der geprüften Hypothesen und eine Darstellung der methodischen, datenspezifischen und technischen Limitationen. Abschließend werden Empfehlungen für den praktischen Einsatz LLM basierter Rechnungsdatenextraktion bei der aifinyo AG formuliert.

9.1 Beantwortung der Forschungsfragen

Zum Abschluss der Arbeit lassen sich die eingangs formulierten Forschungsfragen auf Grundlage der durchgeführten Experimente und Analysen beantworten.

F1: Wie präzise extrahieren Large Language Models strukturierte Daten aus maschinenlesbaren Rechnungs-PDFs?

Die finale Evaluation zeigt, dass aktuelle LLMs strukturierte Rechnungsdaten mit sehr hoher Präzision extrahieren können. Schon mit einfachen Strategien wie dem Baseline Prompt wurden hohe Genauigkeitswerte erreicht. Gemma 3 27B IT erzielte 94,93% $Document\ Accuracy$ und GPT-4.1 erreichte 96,85%. Mit optimierten Strategien konnte die Leistung weiter gesteigert werden. Claude 3 Sonnet erzielte im Few Shot CoT Ansatz 99,36% $Document\ Accuracy$, GPT-4.1 im Few Shot Setup 99,02% und Gemma 3 27B IT im Few Shot Ansatz 97,61%. Diese Werte verdeutlichen, dass LLMs auch bei variierenden Rechnungsformaten konsistente und valide Ergebnisse liefern. Einschränkungen zeigten sich vor allem in Fällen mit fehlerhaften Textlayern oder unklaren Rechnungsfeldern, was den Einfluss der Eingabequalität unterstreicht.

F2: Inwiefern übertreffen LLMs die durch das Rechnungs-OCR von Gini erreichten Ergebnisse, und welche quantifizierbaren Performance-Unterschiede lassen sich empirisch nachweisen?

Die Ergebnisse belegen eine deutliche Überlegenheit der getesteten LLMs gegenüber der bestehenden OCR-Lösung von Gini. Während Gini eine $Document\ Accuracy$ von 87,24% erzielte, lagen die getesteten LLMs je nach Strategie deutlich darüber. Bereits die Baseline Prompts von Claude 3 Sonnet, GPT-4.1 und Gemma 3 27B IT erreichten Werte zwischen 94,93% und 96,85%. In den besten Konfigurationen mit Claude 3 Sonnet im Few-Shot CoT Ansatz und GPT-4.1 im Few-Shot Setup stieg die $Document\ Accuracy$ auf über 99%. Zudem zeigten die LLMs eine höhere Robustheit gegenüber typischen OCR-Fehlern, insbesondere bei Beträgen und Datumsangaben, wodurch die Wahrscheinlichkeit einer vollständig fehlerfreien Dokumentextraktion signifikant erhöht wurde.

9.2 Reflexion der Hypothesen

Die in Abschnitt 8.3 geprüften Hypothesen zeigen, dass LLMs in den durchgeführten Experimenten eine sehr hohe Genauigkeit in der Rechnungsdatenextraktion erreichen konnten. Besonders deutlich wird dies daran, dass bei der besten getesteten Strategie nahezu alle Kreditoren fehlerfrei verarbeitet wurden. Dies verdeutlicht, dass die Modelle nicht nur einzelne Felder zuverlässig erkennen, sondern auch über komplette Layouts hinweg konsistente Ergebnisse erzielen können. Gleichzeitig bestätigte sich die Annahme einer robusten Generalisierbarkeit, da die Modelle auch auf unabhängigen Evaluationsdaten ein stabiles Leistungsbild aufwiesen. Damit wird sichtbar, dass LLMs unter den untersuchten Bedingungen ein belastbares Werkzeug für die Verarbeitung variabler Eingabeformate darstellen.

Die Analyse macht zugleich deutlich, dass die Leistungsfähigkeit nicht uneingeschränkt verallgemeinert werden kann. Die Untersuchung basierte ausschließlich auf historischen Rechnungsdaten der aifinyo AG und nutzte Gini als Vergleichssystem. Aussagen zu anderen OCR-Verfahren oder Datendomänen lassen sich daraus nicht ableiten. Vor diesem Hintergrund sind die Ergebnisse mit $Document\ Accuracy$ Werten von über 99% als ein sehr positives, aber zugleich kontextgebundenes Resultat zu verstehen, das für die aifinyo AG unmittelbar relevant ist, in anderen Szenarien aber überprüft werden müsste.

9.3 Limitationen der Untersuchung

Die Aussagekraft dieser Arbeit ist methodisch dadurch begrenzt, dass nur eine Auswahl an großen Sprachmodellen untersucht wurde, die alle auf klassischen Decoder only Transformer Architekturen basieren. Andere Ansätze wie Encoder Decoder Modelle oder spezialisierte „Document Understanding Architekturen" wie LayoutLM wurden nicht berücksichtigt und könnten in ähnlichen Tests zu abweichenden Ergebnissen führen. Auch der Effekt von zusätzlichem Feintuning wurde nicht untersucht. Darüber hinaus beschränkte sich die Leistungsbewertung auf drei zentrale Rechnungsfelder, nämlich Rechnungsnummer, Rechnungsdatum und Rechnungsbetrag. Aussagen zur Extraktion weiterer Felder wie Positionen, Steuern oder Zahlungsbedingungen lassen sich aus dieser Untersuchung nicht ableiten.

Eine weitere Einschränkung betrifft die verwendeten Daten, die ausschließlich aus dem Kundenbestand der aifinyo AG stammen. Dadurch sind bestimmte Branchen und Rechnungsformate unterrepräsentiert, sodass die Ergebnisse und Prompts nicht ohne Weiteres auf andere Unternehmenskontexte übertragbar sind.

Darüber hinaus ist bei LLMs grundsätzlich das Risiko von Halluzinationen zu berücksichtigen. Auch wenn in den Tests überwiegend konsistente Ergebnisse erzielt wurden, können Modelle in Einzelfällen fehlerhafte oder erfundene Inhalte generieren, die sich nicht unmittelbar aus den Eingabedaten ableiten lassen.

Schließlich ist auch die technische Umsetzung als Einschränkung zu betrachten. Die Evaluationsplattform wurde im Rahmen dieser Masterarbeit von einer Einzelperson entwickelt. Sie ermöglicht eine systematische und reproduzierbare Auswertung, ist jedoch prototypisch und nicht mit dem Reifegrad produktiver Systeme vergleichbar. Aspekte wie API-Limits, Laufzeiten, Integration in bestehende Systeme oder eine detaillierte Kostenbewertung konnten nur eingeschränkt berücksichtigt werden und müssten in einer späteren Umsetzung gesondert untersucht werden.

9.4 Empfehlung für die aifinyo AG

Die Ergebnisse dieser Arbeit zeigen, dass der Einsatz von LLMs in der Rechnungsdatenextraktion ein erhebliches Potenzial bietet. Die getesteten Strategien erreichten in den besten Konfigurationen $Document\ Accuracy$ Werte von über 99%, womit eine deutliche Steigerung der Datenqualität gegenüber der bisherigen OCR-Lösung möglich erscheint. Für den praktischen Einsatz bietet sich ein API-gestützter Einsatz der Modelle an, bei dem die Instruktionen über Prompts definiert werden. Dieses Vorgehen erlaubt eine schnelle Integration und macht den Austausch von Modellen vergleichsweise unkompliziert. Geschlossene Modelle überzeugen durch hohe Leistungsfähigkeit und kalkulierbare Kosten, während offene Modelle insbesondere dann interessant werden, wenn Datenschutzanforderungen oder der Betrieb auf eigener Infrastruktur im Vordergrund stehen.

Neben den in dieser Arbeit untersuchten Kernfeldern sollten für eine produktive Nutzung auch weitere Rechnungsinformationen wie Positionen, Mehrwertsteuer oder Debitoren berücksichtigt werden. Erste Ergebnisse deuten darauf hin, dass stärker zerlegte Extraktionsansätze, also spezifische Prompts für einzelne Felder, hier Vorteile bieten können. Gleichzeitig bleibt die Abhängigkeit von eingebetteten Textlayern eine zentrale Herausforderung. Ein rein auf Textembeddings basierender Ansatz ist zudem risikobehaftet, da fehlerhafte oder gezielt manipulierte Eingaben von den Modellen nicht zuverlässig erkannt werden können. Für einen produktiven Einsatz wären daher zusätzliche Absicherungen notwendig.

Darüber hinaus haben explorative Untersuchungen gezeigt, dass Strategien wie Few-Shot mit layoutspezifischen Beispielen, hybride Verfahren aus eingebettetem Text und OCR oder Consensus Ansätze mit mehreren Modellen weiteres Potenzial bergen. Diese Ansätze wurden nur in kleinem Umfang geprüft und werden im Ausblick in Abschnitt 10 gesondert aufgegriffen.

10 Ausblick und Potentiale

Die Ergebnisse dieser Arbeit zeigen, dass LLMs bereits heute eine sehr hohe Präzision in der Rechnungsdatenextraktion erreichen. Gleichzeitig deuten erste zusätzliche Experimente darauf hin, dass über die getesteten Strategien hinaus weitere methodische und technologische Ansätze Potenzial besitzen. Diese Ergebnisse sind jedoch lediglich explorativer Natur und wurden nicht im Rahmen des in Abschnitt 5.6 beschriebenen standardisierten Versuchsablaufs evaluiert. Der Ausblick skizziert diese Potenziale und zeigt, welche Forschungs- und Entwicklungsrichtungen sich für die aifinyo AG und für die allgemeine Weiterentwicklung der Dokumentenextraktion ergeben.

10.1 Methodische Erweiterungen

Ein Ansatz mit hohem Potenzial ist die Nutzung von Consensus-Strategien. Erste Tests auf dem Developmentdatensatz kombinieren zwei Strategien, nämlich die als beste getestete Few-Shot-CoT-Variante mit Claude 3 Sonnet und einen einfachen CoT-Ansatz mit GPT-4.1, der ausschließlich Bilder der Rechnungen über ein Vision-Modell verarbeitet. Diese Kombination reduziert die Fehlerrate deutlich und erreicht eine $Document\ Accuracy$ von bis zu 99,8% (Abbildung 10.1).

In einem weiteren Versuch werden zehn unterschiedliche Strategien miteinander kombiniert, darunter Varianten mit Claude, GPT und Gemma-3 sowie Verfahren auf Basis von Text, OCR, Vision und Vision+Text. Dieses erweiterte Ensemble erzielt auf dem Developmentdatensatz ein Ergebnis mit nur einem Fehler, sodass 2575 von 2576 Consensus-Ergebnissen korrekt sind. Die Ergebnisse deuten darauf hin, dass die Kombination unterschiedlicher Ansätze zu einer weiteren Verbesserung führen kann. Allerdings geht ein solcher Ansatz zulasten der Abdeckung, da nur Dokumente berücksichtigt werden, bei denen alle Strategien dasselbe Ergebnis liefern, und ist aufgrund des vielfach höheren Ressourcenverbrauchs in der Praxis nur eingeschränkt realistisch. Zudem stellt eine höhere Genauigkeit keine zwingende Folge dar, sondern lediglich eine mögliche Option bei übereinstimmenden Vorhersagen.

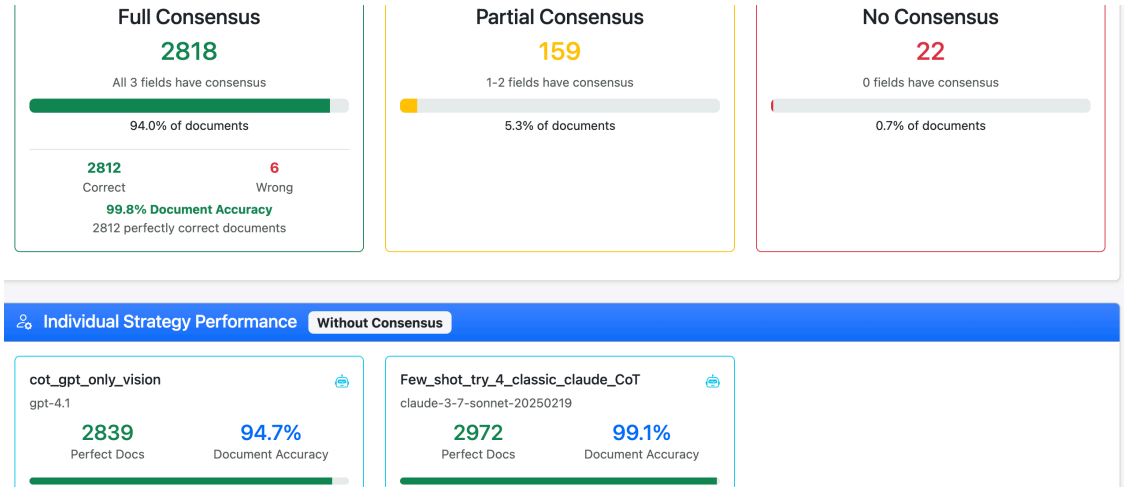


Abbildung 10.1 Ergebnisse der Consensus-Strategie auf dem Developmentdatensatz

Für den praktischen Einsatz sind Few-Shot-Ansätze mit layout-spezifischen Beispielen besonders interessant. Erste Experimente zeigen, dass die Bereitstellung einer einzigen annotierten Rechnung für wiederkehrende Kreditorenlayouts die Extraktion nachfolgender Dokumente deutlich stabilisiert (Abbildung 10.2).

Ein Fallbeispiel verdeutlicht diesen Effekt. Die eingebetteten Daten eines Kreditorenlayouts weisen massive OCR-Fehler auf, etwa „195t22" anstelle von „195/22" bei der Rechnungsnummer oder „11j02022" anstelle von „11.10.2022" beim Rechnungsdatum. Nahezu alle 16 Rechnungen dieses Kreditors enthalten vergleichbare Fehler. Ohne spezifisches Few-Shot-Prompt erreicht die beste getestete Strategie auf den 15 Rechnungen lediglich eine *Document Accuracy* von 13,33%. Wird jedoch eine der fehlerhaften Rechnungen als Beispiel genutzt und den Modellen als Few-Shot-Prompt zur Verfügung gestellt, steigt die *Document Accuracy* auf bis zu 100%. Dieses Ergebnis ist bemerkenswert, weil es darauf hindeutet, dass LLMs in der Lage sind, systematische OCR-Fehler zu erkennen und proaktiv zu korrigieren.

#	Strategy Name	Model	Overall Score	Document Score	Number	Date	Amount	Avg Tokens	Comparisons
1	creditor_2860_few_shot_claude (Best Overall)	claude-3-7-sonnet-20250219	100.00%	100.00% 15/15	100.00% 15/15	100.00% 15/15	100.00% 15/15	1.439 total, 1.395 prompt, 45 completion	15
2	creditor_2860_few_shot_gemma_3_4b	gemma-3-4b-it	86.66%	66.67% 10/15	73.33% 11/15	93.33% 14/15	93.33% 14/15	1.359 total, 1.305 prompt, 54 completion	15
3	creditor_2860_few_shot_gemma_3_27b	gemma-3-27b-it	82.22%	53.33% 8/15	60.00% 9/15	86.67% 13/15	100.00% 15/15	1.359 total, 1.305 prompt, 54 completion	15
4	creditor_2860_few_shot_gpt	gpt-4.1	82.22%	46.67% 7/15	46.67% 7/15	100.00% 15/15	100.00% 15/15	1.091 total, 1.056 prompt, 35 completion	15
5	Few_shot_try_4_classic_claude_CoT	claude-3-7-sonnet-20250219	68.89%	13.33% 2/15	13.33% 2/15	93.33% 14/15	100.00% 15/15	1.761 total, 1.726 prompt, 36 completion	15

Abbildung 10.2 Ergebnisse der Few-Shot-Extraktion für ein einzelnes Kreditorenlayout

10.2 Technologische Perspektiven

Neben methodischen Erweiterungen gewinnen technologische Ansätze an Bedeutung. Multimodale Vision-Modelle, die Text- und Layoutebene gemeinsam verarbeiten, können langfristig die Abhängigkeit von extern erzeugten Textlayern reduzieren. Erste Vergleiche zeigen, dass die reine Vision-Strategie schwächer abschneidet als die textbasierte Variante, während die Kombination von Text- und Vision-Eingaben insgesamt die stabilsten Ergebnisse liefert (Tabelle 10.1).

Auch hybride Verfahren, die unterschiedliche Eingabequellen wie OCR, Embeddings oder Vision kombinieren, zeigen Potenzial. Erste Versuche deuten darauf hin, dass durch den Abgleich verschiedener Quellen typische Problemfälle besser adressiert werden können.

Modell	**Strategie**	***Overall Acc***	***Document Acc***	***Number Acc***	***Date Acc***	***Amount Acc***
GPT-4.1	Text + Vision	98.83%	96.90%	98.03%	99.93%	98.53%
GPT-4.1	Standard CoT	98.74%	96.63%	98.00%	99. 93%	98.30%
GPT-4.1	Only Vision	97.68%	94.66%	97.43%	98.87%	96.73%

Tabelle 10.1 Vergleich Vision-basierter und hybrider Verfahren im Evaluationslauf

10.3 Forschungsperspektiven

Die in den Abschnitten 10.1 und 10.2 skizzierten Erweiterungen verdeutlichen, dass das Potenzial von LLMs für die Rechnungsdatenextraktion längst nicht ausgeschöpft ist. Für die Forschung ergeben sich daraus mehrere Anschlussmöglichkeiten. Ein zentraler Aspekt ist die Evaluation zukünftiger LLM-Generationen im Hinblick auf Genauigkeit, Robustheit und Kosten. Ebenso stellt sich die Frage, wie stabil die Modelle gegenüber Bias und systematischen Verzerrungen in Rechnungsdaten reagieren und welche Verfahren geeignet sind, solche Effekte zu reduzieren.

Darüber hinaus eröffnet sich das Feld des Feintunings und der Domänenadaption. Mit spezialisierten Trainingsdaten könnten zusätzliche Rechnungsfelder wie Positionen, Mehrwertsteuer oder Debitoren gezielt adressiert werden, die in dieser Arbeit nicht systematisch untersucht wurden. Damit ergeben sich konkrete Ansatzpunkte, um die Leistungsfähigkeit der Modelle über die hier getesteten Kernfelder hinaus weiter zu steigern und ihre Eignung für einen produktiven Einsatz zu vertiefen.

11 Fazit

Die vorliegende Arbeit untersuchte die Leistungsfähigkeit von Large Language Models zur Extraktion strukturierter Rechnungsdaten aus maschinenlesbaren PDF-Dokumenten im Anwendungsumfeld der aifinyo AG. Ausgangspunkt war die Hypothese, dass LLMs klassische OCR-basierte Verfahren in Präzision und Robustheit übertreffen können und damit eine zuverlässigere Basis für automatisierte Finanzprozesse schaffen. Im Rahmen systematischer Experimente wurden verschiedene Modelle, Parameterkonfigurationen und Prompting-Strategien evaluiert, ihre Ergebnisse miteinander verglichen und Potenziale für den produktiven Einsatz abgeleitet.

11.1 Erkenntnisse

Die durchgeführten Untersuchungen liefern mehrere zentrale Erkenntnisse:

1. **Deutliche Leistungssteigerung gegenüber OCR**: Alle getesteten LLMs erzielen eine höhere Genauigkeit bei der Extraktion von Rechnungsnummern, Beträgen und Datumsangaben als das bestehende Gini-OCR-System. Die Gini-Ergebnisse basieren dabei auf historisch von Gini prozessierten Rechnungsdaten und erreichen auf Dokumentebene lediglich eine Genauigkeit von 87,24%. Selbst die weniger leistungsstarken LLM-Varianten liegen darüber. Spitzenmodelle wie Claude 3 Sonnet erzielen in der besten Strategie 99,36% $Document\ Accuracy$.
2. **Modellgröße und Architektur beeinflussen Ergebnisqualität**: Proprietäre Modelle (Claude 3 Sonnet, GPT-4.1) liefern konstant die besten Resultate in allen Metriken. Open-Source-Modelle wie Gemma 3 27B-IT erreichen diese Werte zwar nicht vollständig, liegen mit bis zu 97,61% $Document\ Accuracy$ in der Few-Shot-Strategie jedoch deutlich über den aus historischen Daten ermittelten Ergebnissen des Gini-OCR, was ihre Eignung für datenschutzkritische Szenarien unterstreicht.

3. **Prompting-Strategien sind entscheidend**: Zero-Shot-Prompts liefern zwar solide Basisergebnisse, erst optimierte Few-Shot- und CoT-Strategien ermöglichen eine deutlich höhere Genauigkeit. Die optimale Strategie erweist sich jedoch als modellabhängig, sodass keine universelle „beste" Lösung identifiziert werden kann.
4. **Datenqualität als begrenzender Faktor**: Trotz hoher Modellleistungen lassen sich Fehler aus der vorgelagerten OCR- oder PDF-Textgewinnung nicht vollständig kompensieren. Unsaubere oder fehlerhafte OCR-Embeddings im Textlayer können die Extraktionsqualität erheblich beeinträchtigen. Multimodale Ansätze mit direkter Vision-Verarbeitung haben hier tendenziell Vorteile, da sie weniger stark von der Qualität des Textlayers abhängig sind.

Diese Erkenntnisse bestätigen, dass LLMs nicht nur als Ersatz, sondern als qualitative Verbesserung gegenüber klassischen OCR-Lösungen eingesetzt werden können, wenn die Modellwahl und das Prompting sorgfältig auf die jeweiligen Daten und Anforderungen abgestimmt werden. Dabei ist zu berücksichtigen, dass die Gini-Ergebnisse aus historischen Produktionsdaten stammen und somit nicht auf exakt derselben Evaluationsgrundlage wie die LLMs beruhen, sondern als Referenz für die bisherige Systemleistung dienen.

11.2 Praktische Relevanz und Business Impact

Die Ergebnisse dieser Arbeit haben unmittelbare Relevanz für die Automatisierung von Rechnungsprozessen in der aifinyo AG und vergleichbaren Finanzdienstleistungsunternehmen:

- **Qualitätsverbesserung**: Die signifikant höhere Extraktionsgenauigkeit reduziert manuelle Nachbearbeitung, senkt Fehlerraten und verkürzt Durchlaufzeiten im Rechnungsmanagementprozess.
- **Kosteneffizienz**: Weniger manuelle Korrekturen und geringere Fehlerfolgekosten können zu substantiellen Einsparungen führen, selbst wenn die initialen API-Kosten für proprietäre Modelle berücksichtigt werden.
- **Skalierbarkeit**: LLMs ermöglichen eine robuste Verarbeitung auch heterogener und komplexer Belege, was insbesondere bei der Erschließung neuer Kundensegmente mit stark variierenden Rechnungsformaten Vorteile bietet.

- **Zukunftspotenzial**: Die erprobten Strategien und die entwickelte Evaluationsplattform schaffen eine Basis für den Einsatz weiterentwickelter Modelle (z. B. multimodale LLMs), wodurch sich der Automatisierungsgrad künftig weiter steigern lässt.

Die Arbeit legt damit nahe, dass LLM-basierte Extraktionslösungen nicht nur die aktuelle OCR-Lösung von Gini ersetzen, sondern darüber hinaus ein zentrales Element für die zukünftige Automatisierungsstrategie von aifinyo darstellen könnten. Voraussetzung dafür sind eine saubere Trennung von Rechnungs- und Prozessinformationen, ein kontinuierliches Qualitätsmonitoring, die laufende Evaluierung neuer Modellgenerationen auf Basis der in dieser Arbeit entwickelten Methodik sowie ergänzende Kontrollmechanismen, die Manipulationsrisiken in hybriden Szenarien frühzeitig erkennen und abfedern.

Literaturverzeichnis

[1] T. Saout, F. Lardeux, und F. Saubion, „An Overview of Data Extraction From Invoices", *IEEE Access*, Bd. 12, S. 19872–19886, 2024.

[2] Q. Zhang *u. a.*, „Document Parsing Unveiled: Techniques, Challenges, and Prospects for Structured Information Extraction". 28. Oktober 2024. [Online]. Verfügbar unter: https://arxiv.org/abs/2410.21169

[3] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, und G. Neubig, „Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing", *ACM Comput. Surv.*, Bd. 55, Nr. 9, S. 1–35, Jan. 2023.

[4] Sudeep Meduri, „Revolutionizing Customer Service : The Impact of Large Language Models on Chatbot Performance", *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, Bd. 10, Nr. 5, S. 721–730, Nov. 2024.

[5] W. Jiao, W. Wang, J. Huang, X. Wang, S. Shi, und Z. Tu, „Is ChatGPT A Good Translator? Yes With GPT-4 As The Engine". 20. Januar 2023. [Online]. Verfügbar unter: https://arxiv.org/abs/2301.08745

[6] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, und I. Androutsopoulos, „LEGAL-BERT: The Muppets straight out of Law School", in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2020.

[7] A. Arcuri, M. Z. Iqbal, und L. Briand, „Black-Box System Testing of Real-Time Embedded Systems Using Random and Search-Based Testing", in *Lecture Notes in Computer Science*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 95–110.

[8] C. G. Northcutt, A. Athalye, und J. Mueller, „Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks". 26. März 2021. [Online]. Verfügbar unter: https://arxiv.org/abs/2103.14749

[9] S. Ouyang, J. M. Zhang, M. Harman, und M. Wang, „An Empirical Study of the Non-Determinism of ChatGPT in Code Generation", *ACM Trans. Softw. Eng. Methodol.*, Bd. 34, Nr. 2, S. 1–28, Jan. 2025.

[10] H. Bast und C. Korzen, „A Benchmark and Evaluation for Text Extraction from PDF", in *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, IEEE, Juni 2017, S. 1–10.

[11] Aifinyo, „aifinyo - Smart Billment für Ihr Wachstum". 11. März 2021. [Online]. Verfügbar unter: https://www.aifinyo.de/

[12] BaFin, „Merkblatt Factoring". 18. Dezember 2024. [Online]. Verfügbar unter: https://www.bafin.de/SharedDocs/Veroeffentlichungen/DE/Merkblatt/mb_090105_tatbestand_factoring.html

[13] Bundesministerium der Justiz, „Umsatzsteuergesetz, § 14 Ausstellung von Rechnungen". 2. Dezember 2024. [Online]. Verfügbar unter: https://www.gesetze-im-internet.de/ustg_1980/__14.html

[14] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, und M. Zhou, „LayoutLM: Pre-training of Text and Layout for Document Image Understanding", in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA: ACM, Aug. 2020, S. 1192–1200.

[15] H. Zheng, S. Wang, und L. Huang, „A Comprehensive Survey on Document-Level Information Extraction", in *Proceedings of the Workshop on the Future of Event Detection (FuturED)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2024, S. 58–72.
[16] Adobe Systems, „Die Geschichte des PDF: eine Zeitreise." [Online]. Verfügbar unter: https://www.adobe.com/de/acrobat/resources/pdf-timeline.html
[17] H. Krottmaier, „About Problems of PDF", *J. Syst. Cybern. Inform.*, Bd. 4, Dez. 2006.
[18] A. Kumar und L. L. Wang, „Uncovering the New Accessibility Crisis in Scholarly PDFs: Publishing Model and Platform Changes Contribute to Declining Scholarly Document Accessibility in the Last Decade", in *The 26th International ACM SIGACCESS Conference on Computers and Accessibility*, New York, NY, USA: ACM, Okt. 2024, S. 1–16.
[19] N. S. Adhikari und S. Agarwal, „A Comparative Study of PDF Parsing Tools Across Diverse Document Categories". 13. Oktober 2024. [Online]. Verfügbar unter: https://arxiv.org/abs/2410.09871
[20] International Organization for Standardization (ISO), „ISO 32000-2:2020 – Document management — Portable document format — Part 2: PDF 2.0", International Organization for Standardization, Geneva.
[21] Apache Software Foundation, „Apache PDFBox". 1. Mai 2025. [Online]. Verfügbar unter: https://pdfbox.apache.org/
[22] PDFMiner, „Pdfminer.six". 16. April 2025. [Online]. Verfügbar unter: https://pdfminersix.readthedocs.io/en/latest/
[23] PDF-Reader, „GitHub - yob/pdf-reader: The PDF::Reader library implements a PDF parser conforming as much as possible to the PDF specification from Adobe." 13. August 2025. [Online]. Verfügbar unter: https://github.com/yob/pdf-reader
[24] A. Bhattacharyya, A. Tripathi, U. Das, A. Karmakar, A. Pathak, und M. Gupta, „Information Extraction from Visually Rich Documents using LLM-based Organization of Documents into Independent Textual Segments". 18. Mai 2025. [Online]. Verfügbar unter: https://arxiv.org/abs/2505.13535
[25] R. Smith, „An Overview of the Tesseract OCR Engine", in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, IEEE, Sep. 2007, S. 629–633.
[26] M. D. Garris, S. Janet, und W. W. Klein, „Impact of image quality on machine print optical character recognition", National Institute of Standards and Technology, Gaithersburg, MD, 1997.
[27] G. Nagy, T. A. Nartker, und S. V. Rice, „Optical character recognition: an illustrated guide to the frontier", gehalten auf der Electronic Imaging, D. P. Lopresti und J. Zhou, Hrsg., San Jose, CA, Dez. 1999, S. 58–69. doi: 10.1117/12.373511.
[28] T. Blanke, M. Bryant, und M. Hedges, „Open source optical character recognition for historical research", *J. Doc.*, Bd. 68, Nr. 5, S. 659–683, Aug. 2012, doi: 10.1108/00220411211256021.
[29] Camelot, „Camelot: PDF Table Extraction for Humans — Camelot 1.0.9 documentation". 14. Oktober 2019. [Online]. Verfügbar unter: https://camelot-py.readthedocs.io/en/master/

[30] tabula, „Tabula: Extract Tables from PDFs“. 4. Juni 2018. [Online]. Verfügbar unter: https://tabula.technology/
[31] M. Hasan, A. Parameswaran, und A. Cheung, „Benchmarking Extraction of Structured Data from Templatized Documents“, Electrical Engineering and Computer Sciences University of California, Berkeley, Mai 2025.
[32] BMI (Bundesministerium des Innern und für Heimat), „Was ist eine ERechnung?“ [Online]. Verfügbar unter: https://www.e-rechnung-bund.de/e-rechnung/was-ist-eine-e-rechnung/
[33] FeRD (Forum elektronische Rechnung Deutschland), „ZUGFeRD/Factur-X“. [Online]. Verfügbar unter: https://www.ferd-net.de/standards/zugferd
[34] G. Nagy, „Twenty years of document image analysis in PAMI“, *IEEE Trans. Pattern Anal. Mach. Intell.*, Bd. 22, Nr. 1, S. 38–62, 2000.
[35] F. Peng und A. McCallum, „Information extraction from research papers using conditional random fields“, *Inf. Process. Amp Manag.*, Bd. 42, Nr. 4, S. 963–979, Juli 2006.
[36] H. T. Ha, „Recognition of Invoices from Scanned Documents“, in *Proceedings of Recent Advances in Slavonic Natural Language Processing (RASLAN 2017)*, Brno, Czech Republic: Masaryk University, NLP Centre, S. 71–78.
[37] C.-A. Boiangiu, D.-C. Cananau, S. Petrescu, und A. Moldoveanu, „OCR Post Processing Based on Character Pattern Matching“, in *Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium*, Vienna, Austria: DAAAM International Vienna, S. 323–325.
[38] R. B. Palm, O. Winther, und F. Laws, „CloudScan - A configuration-free invoice analysis system using recurrent neural networks“, 2017, *arXiv*. doi: 10.48550/ARXIV.1708.07403.
[39] Gini GmbH, „Gini GmbH“. [Online]. Verfügbar unter: https://gini.net/produkte/extrahieren/gini-smart/#gini-ocr
[40] Klippa, „Klippa DocHorizon - KI-gestützte Dokumentenverarbeitung“. [Online]. Verfügbar unter: https://www.klippa.com/de/dochorizon-de/
[41] T. B. Brown *u. a.*, „Language Models are Few-Shot Learners“. 28. Mai 2020. [Online]. Verfügbar unter: https://arxiv.org/abs/2005.14165
[42] Y. Wang, Q. Yao, J. T. Kwok, und L. M. Ni, „Generalizing from a Few Examples: A Survey on Few-shot Learning“, *ACM Comput. Surv.*, Bd. 53, Nr. 3, S. 1–34, Mai 2021, doi: 10.1145/3386252.
[43] A. Vaswani *u. a.*, „Attention Is All You Need“. 12. Juni 2017. [Online]. Verfügbar unter: https://arxiv.org/abs/1706.03762
[44] A. Radford, K. Narasimhan, T. Salimans, und I. Sutskever, „Improving Language Understanding by Generative Pre-Training“. [Online]. Verfügbar unter: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
[45] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, und I. Sutskever, „Language Models are Unsupervised Multitask Learners“. [Online]. Verfügbar unter: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
[46] Anthropic, „Introducing Claude“. 14. März 2023. [Online]. Verfügbar unter: https://www.anthropic.com/news/introducing-claude

[47] J. Devlin, M.-W. Chang, K. Lee, und K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. 11. Oktober 2018. [Online]. Verfügbar unter: https://arxiv.org/abs/1810.04805
[48] G. Kim *u. a.*, „OCR-free Document Understanding Transformer“. 30. November 2021. [Online]. Verfügbar unter: https://arxiv.org/abs/2111.15664
[49] F. Loukil *u. a.*, „LLM-centric pipeline for information extraction from invoices“, in *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, IEEE, Nov. 2024, S. 569–575.
[50] P. Damodaran, P. Singh, und J. Achankuju, „Zero-shot Task Transfer for Invoice Extraction via Class-aware QA Ensemble“. 13. August 2021. [Online]. Verfügbar unter: https://arxiv.org/abs/2108.06069
[51] A. Akdoğan und M. Kurt, „ExTTNet: A Deep Learning Algorithm for Extracting Table Texts from Invoice Images“. 3. Februar 2024. [Online]. Verfügbar unter: https://arxiv.org/abs/2402.02246
[52] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, und Y. Iwasawa, „Large Language Models are Zero-Shot Reasoners“. 24. Mai 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/2205.11916
[53] T. Z. Zhao, E. Wallace, S. Feng, D. Klein, und S. Singh, „Calibrate Before Use: Improving Few-Shot Performance of Language Models“. 19. Februar 2021. [Online]. Verfügbar unter: https://arxiv.org/abs/2102.09690
[54] J. Wei *u. a.*, „Chain-of-Thought Prompting Elicits Reasoning in Large Language Models“. 28. Januar 2022. [Online]. Verfügbar unter: https://arxiv.org/abs/2201.11903
[55] T. Liu, Z. Chen, Z. Liu, M. Tian, und W. Luo, „Expediting and Elevating Large Language Model Reasoning via Hidden Chain-of-Thought Decoding“. 13. September 2024. [Online]. Verfügbar unter: https://arxiv.org/abs/2409.08561
[56] T. Yang, Z. Li, J. Cao, und C. Xu, „Understanding and Mitigating Hallucination in Large Vision–Language Models via Modular Attribution and Intervention“, in *Proceedings of the International Conference on Learning Representations (ICLR 2025)*, ICLR / OpenReview (du kannst „International Conference on Learning Representations“ schreiben, wenn dein System kein OpenReview kennt).
[57] C. Irugalbandara *u. a.*, „Scaling Down to Scale Up: A Cost-Benefit Analysis of Replacing OpenAI’s LLM with Open Source SLMs in Production“, 2023, doi: 10.48550/ARXIV.2312.14972.
[58] OpenAI *u. a.*, „GPT-4 Technical Report“. 15. März 2023. [Online]. Verfügbar unter: https://arxiv.org/abs/2303.08774
[59] Anthropic, „Introducing the next generation of Claude“. 4. März 2024. [Online]. Verfügbar unter: https://www.anthropic.com/news/claude-3-family
[60] LM Studio, „LM Studio - Download and run LLMs on your computer“. [Online]. Verfügbar unter: https://lmstudio.ai/
[61] Ollama, „Ollama“. [Online]. Verfügbar unter: https://ollama.com/
[62] G. Team *u. a.*, „Gemma: Open Models Based on Gemini Research and Technology“. 13. März 2024. [Online]. Verfügbar unter: https://arxiv.org/abs/2403.08295

[63] H. Touvron *u. a.*, „LLaMA: Open and Efficient Foundation Language Models". 27. Februar 2023. [Online]. Verfügbar unter: https://arxiv.org/abs/2302.13971
[64] A. Q. Jiang *u. a.*, „Mistral 7B". 10. Oktober 2023. [Online]. Verfügbar unter: https://arxiv.org/abs/2310.06825
[65] M. Renze und E. Guven, „The Effect of Sampling Temperature on Problem Solving in Large Language Models". 7. Februar 2024. [Online]. Verfügbar unter: https://arxiv.org/abs/2402.05201
[66] H. Österle, R. Winter, und H. Brenner, *Gestaltungsorientierte Wirtschaftsinformatik: ein Plädoyer für Rigor und Relevanz*. Nürnberg: Infowerk, 2010.
[67] K. Schmid, S. El-Sharkawy, und C. Kröher, „Improving Software Engineering Research through Experimentation Workbenches". 25. Oktober 2021. [Online]. Verfügbar unter: https://arxiv.org/abs/2110.12937
[68] R. Müller und S. Pannasch, „Der Mensch auf Standby: Wie Automatisierung unser Denken und Handeln in technischen Systemen beeinflusst", in *Herbstkonferenz 2015: Arbeitswissenschaft mit Interdisziplinarität und Methodenvielfalt*, Dortmund: Gesellschaft für Arbeitswissenschaft e.V., S. 1–6.
[69] D. M. W. Powers, „Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". 11. Oktober 2020. [Online]. Verfügbar unter: https://arxiv.org/abs/2010.16061
[70] Q. McNemar, „Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages", *Psychometrika*, Bd. 12, Nr. 2, S. 153–157, Juni 1947.
[71] A. Agresti, *An Introduction to Categorical Data Analysis*. Wiley, 2006.
[72] S. Holm, „A Simple Sequentially Rejective Multiple Test Procedure", *Scand. J. Stat.*, Bd. 6, Nr. 2, S. 65–70.
[73] E. B. Wilson, „Probable Inference, the Law of Succession, and Statistical Inference", *J. Am. Stat. Assoc.*, Bd. 22, Nr. 158, S. 209, Juni 1927.
[74] L. D. Brown, T. T. Cai, und A. DasGupta, „Interval Estimation for a Binomial Proportion", *Stat. Sci.*, Bd. 16, Nr. 2, Mai 2001.
[75] Ph. D. C. R. Wolfe, „Automatic Prompt Optimization", *Deep Learn. Focus*, Nov. 2024, [Online]. Verfügbar unter: https://cameronrwolfe.substack.com/p/automatic-prompt-optimization
[76] Y. Xu, Q. An, J. Zhang, P. Li, und Z. Nie, „Hard Sample Aware Prompt-Tuning", in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2023, S. 12356–12369.
[77] Anthropic, „Home".
[78] OpenAI, „OpenAI Platform". [Online]. Verfügbar unter: https://platform.openai.com/docs/api-reference/introduction
[79] ggml-org, „GitHub - ggml-org/llama.cpp: LLM inference in C/C++". 22. August 2025. [Online]. Verfügbar unter: https://github.com/ggml-org/llama.cpp
[80] rubyonrails, „Ruby on Rails: Compress the complexity of modern web apps". [Online]. Verfügbar unter: https://rubyonrails.org/
[81] M. Bendre, B. Sun, D. Zhang, X. Zhou, K. C. Chang, und A. Parameswaran, „DATASPREAD: Unifying Databases and Spreadsheets", in *Proceedings of the VLDB Endowment*, New York: VLDB Endowment, S. 2000–2003.

[82] J. F. Pimentel, L. Murta, V. Braganholo, und J. Freire, „A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks“, in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, IEEE, Mai 2019.
[83] PostgreSQL, „8.14. JSON Types“. 14. August 2025. [Online]. Verfügbar unter: https://www.postgresql.org/docs/current/datatype-json.html
[84] S. V. Salunke und A. Ouda, „A Performance Benchmark for the PostgreSQL and MySQL Databases“, *Future Internet*, Bd. 16, Nr. 10, S. 382, Okt. 2024.
[85] PostgreSQL, „41.10. Trigger Functions“. 14. August 2025. [Online]. Verfügbar unter: https://www.postgresql.org/docs/17/plpgsql-trigger.html
[86] B. Shaik und D. K. Chemuduru, *Procedural Programming with PostgreSQL PL/pgSQL: Design Complex*. Berkeley, CA: Apress, 2023.
[87] jsvine, „GitHub - jsvine/pdfplumber: Plumb a PDF for detailed information about each char, rectangle, line, et cetera — and easily extract text and tables.“ 19. Juli 2025. [Online]. Verfügbar unter: https://github.com/jsvine/pdfplumber
[88] pymupdf, „GitHub - pymupdf/PyMuPDF: PyMuPDF is a high performance Python library for data extraction, analysis, conversion & manipulation of PDF (and other) documents.“ 22. Juli 2024. [Online]. Verfügbar unter: https://github.com/pymupdf/PyMuPDF
[89] pypdfium2-team, „GitHub - pypdfium2-team/pypdfium2: Python bindings to PDFium, reasonably cross-platform.“ 27. August 2025. [Online]. Verfügbar unter: https://github.com/pypdfium2-team/pypdfium2
[90] sidekiq, „GitHub - sidekiq/sidekiq: Simple, efficient background processing for Ruby“. 6. März 2025. [Online]. Verfügbar unter: https://github.com/sidekiq/sidekiq
[91] Redis, „Redis“. 10. Februar 2025. [Online]. Verfügbar unter: https://redis.io/
[92] M. Fowler, „Inversion of Control Containers and the Dependency Injection pattern“. 23. Januar 2004. [Online]. Verfügbar unter: https://martinfowler.com/articles/injection.html
[93] A. R. Gosthipaty, M. Noyan, P. Cuenca, und V. Srivastav, „Welcome Gemma 3: Google’s all new multimodal, multilingual, long context open LLM“. 12. März 2025. [Online]. Verfügbar unter: https://huggingface.co/blog/gemma3?utm_source=chatgpt.com
[94] J. Yuan *u. a.*, „Give Me FP32 or Give Me Death? Challenges and Solutions for Reproducible Reasoning“. 11. Juni 2025. [Online]. Verfügbar unter: https://arxiv.org/abs/2506.09501
[95] J. Nielsen, „Response Times: The 3 Important Limits“, *Nielsen Norman Group*, Jan. 1993, [Online]. Verfügbar unter: https://www.nngroup.com/articles/response-times-3-important-limits

Anhang

Anhang 1: Prompts

Anhang 1.1: JSON-Struktur einer Rechnung

```
{
  "invoiceNumber": "String",
  "invoiceDate": " String ",
  "invoiceAmount": "Float",
  "invoiceAmountUst": [
    {
      "rate": "String",
      "amount": "Float"
    }
  ],
  "targetDays": "Integer",
  "assignmentNote": "Array|null",
  "taxExemptionReason": "String|null",
  "sender": {
    "name": "String",
    "street": "String",
    "streetNumber": "String|null",
    "zip": "String",
    "city": "String",
    "tradeRegisterNumber": "String|null",
    "taxNumber": "String|null",
    "UstID": "String|null"
  },
  "recipient": {
    "extern_id": "String|null",
    "name": "String",
    "poBox": "String|null",
    "street": "String",
    "streetNumber": "String|null",
    "zip": "String",
    "city": "String"
  }
}

```

Anhang 1.2: Ausgangsprompt

```
System
You are an AI that extracts invoice information and returns it as valid JSON.
- Do NOT include markdown formatting such as ```json.
- Do NOT include extra text or explanations.
- invoiceAmount is always Brutto
- Respond with raw JSON only. Dates like invoiceDate must be iso date strings (YYYY-MM-DD)
- You must validate the datatypes. This will be validated. String must be always a String
and Double always a Double.

User
Extract the following invoice details and return ONLY based on the following JSON struc-
ture:

{ \"invoiceNumber\": \"String\",
  \"invoiceDate\": \"Date\",
  \"invoiceAmount\": \"Double\",
  \"invoiceAmountUst\": [ { \"rate\": \"19%\", \"amount\": \"Double\" } ],
  \"targetDays\": \"Integer\",
  \"assignmentNote\": \"String\",
  \"taxExemptionReason\": \"String\",
  \"sender\": {
          \"name\": \"String\",
          \"street\": \"String\",
          \"streetNumber\": \"String\",
          \"zip\": \"String\",
          \"city\": \"String\",
          \"tradeRegisterNumber\": \"String\",
          \"taxNumber\": \"String\",
          \"UstID\": \"String\" },
  \"recipient\": {
          \"extern_id\": \"String\",
          \"name\": \"String\",
          \"poBox\": \"String\",
          \"street\": \"String\",
          \"streetNumber\": \"String\",
          \"zip\": \"String\",
          \"city\": \"String\" },
  \"bankDetails\": { \"iban\": \"String\", \"bic\": \"String\" },
  \"positions\": [ {
          \"index\": \"Integer\",
          \"itemCount\": \"Double\",
          \"description\": \"String\",
          \"itemAmount\": \"Double\",
          \"amount\": \"Double\",
          \"currency\": \"String\" } ] }

Process the following extracted text and return ONLY the JSON:

$text
```

Anhang 1.3: Verkürzter Prompt

```
System

You are an AI that extracts invoice information and returns it as valid JSON.
- Do NOT include markdown formatting such as ```json.
- Do NOT include extra text or explanations.
- invoiceAmount is always Brutto
- Respond with raw JSON only. Dates like invoiceDate must be iso date strings (YYYY-MM-DD)
- You must validate the datatypes. This will be validated. String must be always a String
and Double always a

User

Extract the following invoice details and return ONLY based on the following JSON struc-
ture: {
  \"invoiceNumber\": \"String\",
  \"invoiceDate\": \"Date\",
  \"invoiceAmount\": \"Double\"} ] }

Process the following extracted text and return ONLY the JSON: $text
```

Anhang 2: SQL Querys

Anhang 2.1: Trigger Function

Trigger

```
 1. create function trigger_recalculate_comparisons() returns trigger
 2.     language plpgsql
 3. as
 4. $$
 5. BEGIN
 6.     IF TG_OP = 'INSERT' OR (TG_OP = 'UPDATE' AND (NEW.content)::text IS DISTINCT FROM
(OLD.content)::text) THEN
 7.         IF NEW.extraction_type::text IN ('llm', 'gini', 'infact') THEN
 8.             PERFORM recalculate_extraction_comparison(NEW.id);
 9.         END IF;
10.     END IF;
11.
12.     RETURN NEW;
13. END;
14. $$;
15.
16. alter function trigger_recalculate_comparisons() owner to postgres;
```

Function recalculate_document_extraction_comparisons

```
 1. create function recalculate_document_extraction_comparisons(document_id_param integer)
returns void
 2.     language plpgsql
 3. as
 4. $$
 5.       DECLARE
 6.           extraction_record RECORD;
 7.       BEGIN
 8.           -- Recalculate comparisons for all LLM and GINI extractions for this
document
 9.           FOR extraction_record IN
10.               SELECT id FROM extractions
11.               WHERE document_id = document_id_param
12.                 AND extraction_type IN ('LLM'::extractiontype, 'GINI'::extractiontype)
13.           LOOP
14.               PERFORM recalculate_extraction_comparison(extraction_record.id);
15.           END LOOP;
16.
17.           -- Recalculate comparisons for all INFACT extractions for this document
18.           FOR extraction_record IN
19.               SELECT id FROM extractions
20.               WHERE document_id = document_id_param
21.                 AND extraction_type = 'INFACT'::extractiontype
22.           LOOP
23.               PERFORM recalculate_extraction_comparison(extraction_record.id);
24.           END LOOP;
25.       END;
26.       $$;
27.
28. alter function recalculate_document_extraction_comparisons(integer) owner to postgres;
```

Function recalculate_extraction_comparison

```
 1. create function public.recalculate_extraction_comparison(extraction_id integer) re-
turns void
 2.     language plpgsql
 3. as
 4. $$
 5.       DECLARE
 6.           extraction_row RECORD;
 7.           other RECORD;
 8.           cmp_id INTEGER;
 9.
10.           llm_invoice_number TEXT;
11.           infact_invoice_number TEXT;
12.           llm_invoice_amount TEXT;
13.           infact_invoice_amount TEXT;
14.           llm_invoice_date TEXT;
15.           infact_invoice_date TEXT;
16.       BEGIN
17.           SELECT * INTO extraction_row FROM extractions WHERE id = extraction_id;
18.
19.           -- LLM oder GINI gegen INFACT vergleichen
20.           IF extraction_row.extraction_type IN ('LLM'::extractiontype, 'GINI'::extrac-
tiontype) THEN
21.               SELECT * INTO other FROM extractions
22.               WHERE document_id = extraction_row.document_id
23.                 AND extraction_type = 'INFACT'::extractiontype
24.               ORDER BY created_at DESC
25.               LIMIT 1;
26.
27.               IF other.id IS NULL THEN RETURN; END IF;
28.
29.               IF jsonb_typeof(extraction_row.content::jsonb) = 'array' THEN
30.                   llm_invoice_number := extraction_row.content::jsonb -> 0 ->> 'in-
voiceNumber';
31.                   llm_invoice_amount := try_parse_float(extraction_row.content::jsonb
-> 0 ->> 'invoiceAmount')::TEXT;
32.                   llm_invoice_date   := extraction_row.content::jsonb -> 0 ->> 'in-
voiceDate';
33.               ELSE
34.                   llm_invoice_number := extraction_row.content::jsonb ->> 'invoiceNum-
ber';
35.                   llm_invoice_amount := try_parse_float(extraction_row.content::jsonb
->> 'invoiceAmount')::TEXT;
36.                   llm_invoice_date   := extraction_row.content::jsonb ->> 'invoiceDa-
te';
37.               END IF;
38.
39.               IF jsonb_typeof(other.content::jsonb) = 'array' THEN
40.                   infact_invoice_number := other.content::jsonb -> 0 ->> 'invoiceNum-
ber';
41.                   infact_invoice_amount := try_parse_float(other.content::jsonb -> 0 -
>> 'invoiceAmount')::TEXT;
42.                   infact_invoice_date   := other.content::jsonb -> 0 ->> 'invoiceDa-
te';
43.               ELSE
44.                   infact_invoice_number := other.content::jsonb ->> 'invoiceNumber';
45.                   infact_invoice_amount := try_parse_float(other.content::jsonb ->>
'invoiceAmount')::TEXT;
46.                   infact_invoice_date   := other.content::jsonb ->> 'invoiceDate';
47.               END IF;
48.
49.               SELECT id INTO cmp_id FROM extraction_comparisons
50.               WHERE llm_extraction_id = extraction_row.id AND infact_extraction_id =
other.id;
51.
52.               IF cmp_id IS NOT NULL THEN
53.                   UPDATE extraction_comparisons SET
54.                       invoice_number_match = (llm_invoice_number = infact_invoice_num-
ber),
```

```
55.                         invoice_number_llm_value = llm_invoice_number,
56.                         invoice_number_infact_value = infact_invoice_number,
57.                         invoice_amount_match = (llm_invoice_amount = infact_in-
voice_amount),
58.                         invoice_amount_llm_value = llm_invoice_amount,
59.                         invoice_amount_infact_value = infact_invoice_amount,
60.                         invoice_date_match = (llm_invoice_date = infact_invoice_date),
61.                         invoice_date_llm_value = llm_invoice_date,
62.                         invoice_date_infact_value = infact_invoice_date,
63.                         updated_at = NOW()
64.                     WHERE id = cmp_id;
65.                 ELSE
66.                     INSERT INTO extraction_comparisons (
67.                         llm_extraction_id, infact_extraction_id,
68.                         invoice_number_match, invoice_number_llm_value, invoice_num-
ber_infact_value,
69.                         invoice_amount_match, invoice_amount_llm_value, in-
voice_amount_infact_value,
70.                         invoice_date_match, invoice_date_llm_value, invoice_date_in-
fact_value,
71.                         created_at, updated_at
72.                     ) VALUES (
73.                         extraction_row.id, other.id,
74.                         (llm_invoice_number = infact_invoice_number), llm_invoice_num-
ber, infact_invoice_number,
75.                         (llm_invoice_amount = infact_invoice_amount), llm_in-
voice_amount, infact_invoice_amount,
76.                         (llm_invoice_date = infact_invoice_date), llm_invoice_date, in-
fact_invoice_date,
77.                         NOW(), NOW()
78.                     );
79.                 END IF;
80.
81.             ELSIF extraction_row.extraction_type = 'INFACT'::extractiontype THEN
82.                 FOR other IN
83.                     SELECT * FROM extractions
84.                     WHERE document_id = extraction_row.document_id
85.                       AND extraction_type IN ('LLM'::extractiontype, 'GINI'::extraction-
type)
86.                 LOOP
87.                     PERFORM recalculate_extraction_comparison(other.id);
88.                 END LOOP;
89.             END IF;
90.         END;
91.         $$;
92.
93. alter function public.recalculate_extraction_comparison(integer) owner to postgres;
```

Anhang 2.2: prediction_results_polished View

```
 1. create materialized view public.prediction_results_polished as
 2. SELECT documents.id,
 3.        e_t.content -> 'text'::text                                          AS text,
 4.        (e_g.content -> 'invoiceNumber'::text)::text                         AS gini_in-
voicenumber,
 5.        (e_i.content -> 'invoiceNumber'::text)::text                         AS in-
fact_invoicenumber,
 6.        CASE
 7.            WHEN ((e_g.content -> 'invoiceNumber'::text)::text) = ((e_i.content -> 'in-
voiceNumber'::text)::text)
 8.                THEN 'GOOD'::text
 9.            ELSE 'BAD'::text
10.            END                                                              AS num-
ber_prediction,
11.        (e_g.content -> 'invoiceAmount'::text)::text                         AS gini_in-
voiceamount,
12.        (e_i.content -> 'invoiceAmount'::text)::text                         AS in-
fact_invoiceamount,
13.        CASE
14.            WHEN (e_g.content ->> 'invoiceAmount'::text) ~ '^\d+(\.\d+)?$'::text AND
15.                 (e_i.content ->> 'invoiceAmount'::text) ~ '^\d+(\.\d+)?$'::text AND
16.                 (e_g.content ->> 'invoiceAmount'::text) <> 'null'::text AND
17.                 (((e_g.content ->> 'invoiceAmount'::text)::double precision)::text) =
18.                 (((e_i.content ->> 'invoiceAmount'::text)::double precision)::text)
THEN 'GOOD'::text
19.            ELSE 'BAD'::text
20.            END                                                              AS
amount_prediction,
21.        e_g.content -> 'invoiceDate'::text                                   AS gini_in-
voicedate,
22.        e_i.content -> 'invoiceDate'::text                                   AS in-
fact_invoicedate,
23.        CASE
24.            WHEN ((e_g.content -> 'invoiceDate'::text)::text) = ((e_i.content -> 'in-
voiceDate'::text)::text) THEN 'GOOD'::text
25.            ELSE 'BAD'::text
26.            END                                                              AS
date_prediction,
27.        e_g.content -> 'targetDays'::text                                    AS
gini_targetdays,
28.        e_i.content -> 'targetDays'::text                                    AS in-
fact_targetdays,
29.        CASE
30.            WHEN ((e_g.content -> 'targetDays'::text)::text) = ((e_i.content -> 'tar-
getDays'::text)::text) THEN 'GOOD'::text
31.            ELSE 'BAD'::text
32.            END                                                              AS target-
days_prediction,
33.        e_g.content -> 'invoiceAmountUst'::text                              AS gini_in-
voiceamountust,
34.        ((e_i.content -> 'invoiceAmountUst'::text) -> 0) ->> 'amount'::text AS in-
fact_invoiceamountust,
35.        CASE
36.            WHEN ((e_g.content -> 'invoiceAmountUst'::text)::text) =
37.                 (((e_i.content -> 'invoiceAmountUst'::text) -> 0) ->> 'amount'::text)
THEN 'GOOD'::text
38.            ELSE 'BAD'::text
39.            END                                                              AS ust_pre-
diction,
40.        (e_g.content -> 'sender'::text) ->> 'name'::text                     AS
gini_sender_name,
41.        (e_i.content -> 'sender'::text) ->> 'name'::text                     AS in-
fact_sender_name,
42.        CASE
43.            WHEN ((e_g.content -> 'sender'::text) ->> 'name'::text) = ((e_i.content ->
'sender'::text) ->> 'name'::text)
```

```
44.                THEN 'GOOD'::text
45.            ELSE 'BAD'::text
46.            END                                                    AS sen-
der_prediction
47. FROM documents
48.          JOIN extractions e_g ON documents.id = e_g.document_id AND e_g.extrac-
tion_type = 'GINI'::extractiontype
49.          JOIN extractions e_t
50.               ON documents.id = e_t.document_id AND e_t.extraction_type =
'EXTRACT_TEXT'::extractiontype AND
51.                  length((e_t.content -> 'text'::text)::text) > 25 AND
52.                  ((e_t.content -> 'text'::text)::text) !~~ '(cid:%'::text AND
53.                  ((e_t.content -> 'text'::text)::text) !~~ '%(cid:%'::text AND
54.                  (length((e_t.content -> 'text'::text)::text) -
55.                   length(replace((e_t.content -> 'text'::text)::text, '\ufffd'::text,
''::text))) < 30 AND
56.                  ((e_t.content -> 'text'::text)::text) !~~ '%tundenzettel%'::text AND
57.                  ((e_t.content -> 'text'::text)::text) !~~ '%Negatives Abrechnungs-
konto%'::text AND
58.                  length((e_t.content -> 'text'::text)::text) < 10000
59.          JOIN extractions e_i ON documents.id = e_i.document_id AND e_i.extrac-
tion_type = 'INFACT'::extractiontype
60. WHERE ((e_i.content -> 'sender'::text) ->> 'name'::text) IS NOT NULL
61.   AND ((e_i.content -> 'sender'::text) ->> 'name'::text) !~~ 'aifinyo%'::text;
62.
63. alter materialized view public.prediction_results_polished owner to postgres;
```

Anhang 2.3: Developmentset Materialized View

```
create materialized view public.test_set_3k_complex as
WITH eval_set_ids AS (SELECT eval_set_10k.id
                      FROM eval_set_10k),
     base AS (SELECT prediction_results_polished2.id,
                     prediction_results_polished2.text,
                     prediction_results_polished2.gini_invoicenumber,
                     prediction_results_polished2.infact_invoicenumber,
                     prediction_results_polished2.number_prediction,
                     prediction_results_polished2.gini_invoiceamount,
                     prediction_results_polished2.infact_invoiceamount,
                     prediction_results_polished2.amount_prediction,
                     prediction_results_polished2.gini_invoicedate,
                     prediction_results_polished2.infact_invoicedate,
                     prediction_results_polished2.date_prediction,
                     prediction_results_polished2.gini_targetdays,
                     prediction_results_polished2.infact_targetdays,
                     prediction_results_polished2.targetdays_prediction,
                     prediction_results_polished2.gini_invoiceamountust,
                     prediction_results_polished2.infact_invoiceamountust,
                     prediction_results_polished2.ust_prediction,
                     prediction_results_polished2.gini_sender_name,
                     prediction_results_polished2.infact_sender_name,
                     prediction_results_polished2.sender_prediction,
                     row_number() OVER () AS internal_order
              FROM prediction_results_polished2
              WHERE NOT (prediction_results_polished2.id IN (SELECT eval_set_ids.id
                                                             FROM eval_set_ids))),
     number_good AS (SELECT base.id,
                            'number_good'::text AS label,
                            base.internal_order
                     FROM base
                     WHERE base.number_prediction = 'GOOD'::text),
     number_bad AS (SELECT base.id,
                           'number_bad'::text AS label,
                           base.internal_order
                    FROM base
                    WHERE base.number_prediction = 'BAD'::text),
     amount_good AS (SELECT base.id,
                            'amount_good'::text AS label,
                            base.internal_order
                     FROM base
                     WHERE base.amount_prediction = 'GOOD'::text),
     amount_bad AS (SELECT base.id,
                           'amount_bad'::text AS label,
                           base.internal_order
                    FROM base
                    WHERE base.amount_prediction = 'BAD'::text),
     date_good AS (SELECT base.id,
                          'date_good'::text AS label,
                          base.internal_order
                   FROM base
                   WHERE base.date_prediction = 'GOOD'::text),
     date_bad AS (SELECT base.id,
                         'date_bad'::text AS label,
                         base.internal_order
                  FROM base
                  WHERE base.date_prediction = 'BAD'::text),
     unioned AS (SELECT number_good.id,
                        number_good.label,
                        number_good.internal_order
                 FROM number_good
                 UNION ALL
                 SELECT number_bad.id,
                        number_bad.label,
                        number_bad.internal_order
                 FROM number_bad
```

```
 67.                   UNION ALL
 68.                   SELECT amount_good.id,
 69.                          amount_good.label,
 70.                          amount_good.internal_order
 71.                   FROM amount_good
 72.                   UNION ALL
 73.                   SELECT amount_bad.id,
 74.                          amount_bad.label,
 75.                          amount_bad.internal_order
 76.                   FROM amount_bad
 77.                   UNION ALL
 78.                   SELECT date_good.id,
 79.                          date_good.label,
 80.                          date_good.internal_order
 81.                   FROM date_good
 82.                   UNION ALL
 83.                   SELECT date_bad.id,
 84.                          date_bad.label,
 85.                          date_bad.internal_order
 86.                   FROM date_bad),
 87.      prioritized_ids AS (SELECT unioned.id,
 88.                                 unioned.label,
 89.                                 unioned.internal_order,
 90.                                 row_number()
 91.                                 OVER (PARTITION BY unioned.id ORDER BY unioned.inter-
nal_order) AS priority_rank
 92.                          FROM unioned),
 93.      joined_with_sender AS (SELECT pr_1.id,
 94.                                    pr_1.text,
 95.                                    pr_1.gini_invoicenumber,
 96.                                    pr_1.infact_invoicenumber,
 97.                                    pr_1.number_prediction,
 98.                                    pr_1.gini_invoiceamount,
 99.                                    pr_1.infact_invoiceamount,
100.                                    pr_1.amount_prediction,
101.                                    pr_1.gini_invoicedate,
102.                                    pr_1.infact_invoicedate,
103.                                    pr_1.date_prediction,
104.                                    pr_1.gini_targetdays,
105.                                    pr_1.infact_targetdays,
106.                                    pr_1.targetdays_prediction,
107.                                    pr_1.gini_invoiceamountust,
108.                                    pr_1.infact_invoiceamountust,
109.                                    pr_1.ust_prediction,
110.                                    pr_1.gini_sender_name,
111.                                    pr_1.infact_sender_name,
112.                                    pr_1.sender_prediction,
113.                                    pi.label,
114.                                    pi.internal_order
115.                             FROM prioritized_ids pi
116.                                      JOIN prediction_results_polished2 pr_1 ON
pr_1.id = pi.id
117.                             WHERE pi.priority_rank = 1),
118.      sender_diverse AS (SELECT t.id,
119.                                t.text,
120.                                t.gini_invoicenumber,
121.                                t.infact_invoicenumber,
122.                                t.number_prediction,
123.                                t.gini_invoiceamount,
124.                                t.infact_invoiceamount,
125.                                t.amount_prediction,
126.                                t.gini_invoicedate,
127.                                t.infact_invoicedate,
128.                                t.date_prediction,
129.                                t.gini_targetdays,
130.                                t.infact_targetdays,
131.                                t.targetdays_prediction,
132.                                t.gini_invoiceamountust,
133.                                t.infact_invoiceamountust,
134.                                t.ust_prediction,
```

```
135.                                t.gini_sender_name,
136.                                t.infact_sender_name,
137.                                t.sender_prediction,
138.                                t.label,
139.                                t.internal_order,
140.                                t.sender_rank
141.                           FROM (SELECT joined_with_sender.id,
142.                                        joined_with_sender.text,
143.                                        joined_with_sender.gini_invoicenumber,
144.                                        joined_with_sender.infact_invoicenumber,
145.                                        joined_with_sender.number_prediction,
146.                                        joined_with_sender.gini_invoiceamount,
147.                                        joined_with_sender.infact_invoiceamount,
148.                                        joined_with_sender.amount_prediction,
149.                                        joined_with_sender.gini_invoicedate,
150.                                        joined_with_sender.infact_invoicedate,
151.                                        joined_with_sender.date_prediction,
152.                                        joined_with_sender.gini_targetdays,
153.                                        joined_with_sender.infact_targetdays,
154.                                        joined_with_sender.targetdays_prediction,
155.                                        joined_with_sender.gini_invoiceamountust,
156.                                        joined_with_sender.infact_invoiceamountust,
157.                                        joined_with_sender.ust_prediction,
158.                                        joined_with_sender.gini_sender_name,
159.                                        joined_with_sender.infact_sender_name,
160.                                        joined_with_sender.sender_prediction,
161.                                        joined_with_sender.label,
162.                                        joined_with_sender.internal_order,
163.                                        row_number()
164.                                        OVER (PARTITION BY joined_with_sender.label, jo-
ined_with_sender.sender_prediction ORDER BY joined_with_sender.internal_order) AS sen-
der_rank
165.                                 FROM joined_with_sender) t
166.                           WHERE t.sender_rank = 1),
167.      fallback_fill AS (SELECT t.id,
168.                                t.text,
169.                                t.gini_invoicenumber,
170.                                t.infact_invoicenumber,
171.                                t.number_prediction,
172.                                t.gini_invoiceamount,
173.                                t.infact_invoiceamount,
174.                                t.amount_prediction,
175.                                t.gini_invoicedate,
176.                                t.infact_invoicedate,
177.                                t.date_prediction,
178.                                t.gini_targetdays,
179.                                t.infact_targetdays,
180.                                t.targetdays_prediction,
181.                                t.gini_invoiceamountust,
182.                                t.infact_invoiceamountust,
183.                                t.ust_prediction,
184.                                t.gini_sender_name,
185.                                t.infact_sender_name,
186.                                t.sender_prediction,
187.                                t.label,
188.                                t.internal_order,
189.                                t.group_rank
190.                           FROM (SELECT joined_with_sender.id,
191.                                        joined_with_sender.text,
192.                                        joined_with_sender.gini_invoicenumber,
193.                                        joined_with_sender.infact_invoicenumber,
194.                                        joined_with_sender.number_prediction,
195.                                        joined_with_sender.gini_invoiceamount,
196.                                        joined_with_sender.infact_invoiceamount,
197.                                        joined_with_sender.amount_prediction,
198.                                        joined_with_sender.gini_invoicedate,
199.                                        joined_with_sender.infact_invoicedate,
200.                                        joined_with_sender.date_prediction,
201.                                        joined_with_sender.gini_targetdays,
202.                                        joined_with_sender.infact_targetdays,
```

```
203.                                    joined_with_sender.targetdays_prediction,
204.                                    joined_with_sender.gini_invoiceamountust,
205.                                    joined_with_sender.infact_invoiceamountust,
206.                                    joined_with_sender.ust_prediction,
207.                                    joined_with_sender.gini_sender_name,
208.                                    joined_with_sender.infact_sender_name,
209.                                    joined_with_sender.sender_prediction,
210.                                    joined_with_sender.label,
211.                                    joined_with_sender.internal_order,
212.                                    row_number()
213.                                    OVER (PARTITION BY joined_with_sender.label ORDER
BY joined_with_sender.internal_order) AS group_rank
214.                             FROM joined_with_sender
215.                             WHERE NOT (joined_with_sender.id IN (SELECT sender_di-
verse.id
216.                                                                  FROM sender_di-
verse))) t
217.                       WHERE t.group_rank <= 500),
218.      limited_ids AS (SELECT sender_diverse.id
219.                      FROM sender_diverse
220.                      UNION
221.                      SELECT fallback_fill.id
222.                      FROM fallback_fill),
223.      random_fill AS (SELECT prediction_results_polished2.id
224.                      FROM prediction_results_polished2
225.                      WHERE NOT (prediction_results_polished2.id IN (SELECT
eval_set_10k.id
226.                                                                     FROM
eval_set_10k))
227.                        AND NOT (prediction_results_polished2.id IN (SELECT limi-
ted_ids.id
228.                                                                     FROM limi-
ted_ids))
229.                      ORDER BY (random())
230.                      LIMIT 1000),
231.      final_3000 AS (SELECT combined.id
232.                     FROM (SELECT limited_ids.id
233.                           FROM limited_ids
234.                           UNION ALL
235.                           SELECT random_fill.id
236.                           FROM random_fill) combined
237.                     ORDER BY (random())
238.                     LIMIT 3000)
239. SELECT DISTINCT ON (pr.id) pr.id,
240.                            pr.text,
241.                            pr.gini_invoicenumber,
242.                            pr.infact_invoicenumber,
243.                            pr.number_prediction,
244.                            pr.gini_invoiceamount,
245.                            pr.infact_invoiceamount,
246.                            pr.amount_prediction,
247.                            pr.gini_invoicedate,
248.                            pr.infact_invoicedate,
249.                            pr.date_prediction,
250.                            pr.gini_targetdays,
251.                            pr.infact_targetdays,
252.                            pr.targetdays_prediction,
253.                            pr.gini_invoiceamountust,
254.                            pr.infact_invoiceamountust,
255.                            pr.ust_prediction,
256.                            pr.gini_sender_name,
257.                            pr.infact_sender_name,
258.                            pr.sender_prediction
259. FROM prediction_results_polished2 pr
260.          JOIN final_3000 f ON pr.id = f.id
261. ORDER BY pr.id;
262.
263. alter materialized view public.test_set_3k_complex owner to postgres;
```

Anhang 3: Tabellen

Anhang 3.1: Parsing-Analyse

Strategie	Direct JSON	Array Unwrap	Regex Fallback	Text Fallback	Parse Failures
Gemma 3 1B Naive	89,8%	0%	0,7%	0,2%	9,3%
Gemma 3 4B Naive	98,9%	0%	0,1%	0%	1%
Gemma 3 12B Naive	99,9%	0%	0%	0%	0,1%
Gemma 3 27B Naive	99,4%	0%	0%	0%	0,6%
GPT Naive	99,9%	0%	0%	0%	0,1%
Claude Naive	99,1%	0%	0%	0%	0,9%
Gemma 3 4B eval	59%	40,9%	0%	0%	0,1%
Gemma 3 27B eval	20%	79,7%	0%	0%	0,3%
GPT eval	100%	0%	0,2%	0%	0%
Claude eval	100%	0%	0%	0%	0%

Anhang 3.2: Ausgangs-Performance mit Baseline Prompt

Model	*Overall Acc*	*Document Acc*	*Number Acc*	*Date Acc*	*Amount Acc*
GPT-4.1	97,70%	94,07%	97,83%	98,80%	96,47%
Claude 3 Sonnet (20250219)	97,35%	93,87%	98,40%	98,67%	94,97%
Gemma 3 27B-IT	97,12%	93,00%	97,97%	99,10%	94,30%
Gemma 3 12B-IT	96,04%	89,47%	96,63%	97,23%	94,27%
Gemma 3 4B-IT	82,68%	63,03%	81,30%	88,50%	78,23%
Gemma 3 1B-IT	54,25%	24,97%	46,72%	69,46%	45,58%
Gini	84,01%	57,87%	85,70%	83,17%	83,17%

Anhang 3.3: Accuracy und Laufzeit – Baseline vs reduzierter Prompt

Model	Felder	Accuracy	Duration AVG	Completion Token AVG
gemma-3-4b-it	Abbildung 5.1	82,68%	15,51 s 1,6 s – 91,55 s σ: 8,96 s	598 225 - 8191 σ: 379
gemma-3-27b-it	Abbildung 5.1	94,7%	14,13 s 5,18 s – 118,31 s σ: 10,47 s	596 200 - 4726 σ: 412
gemma-3-4b-it	Rechnungsnummer, Rechnungsdatum, Rechnungsbetrag	91,12%	0,45 s 0,32 s – 9,75 s σ: 0,31 s	58 48 - 1529 σ: 48
gemma-3-27b-it	Rechnungsnummer, Rechnungsdatum, Rechnungsbetrag	96,8%	1,73 s 0,88 s – 49,87 s σ: 0,61 s	54 37 - 199 σ: 8

Anhang 3.4: Ergebnisübersicht der Zero-Shot-Optimierungsstrategien

Modell	Strategie	*Overall Acc*	*Document Acc*	*Number Acc*	*Date Acc*	*Amount Acc*
Claude 3 Sonnet	restriktiv	98,62%	96,37%	99,37%	99,43%	97,07%
	einfach	97,60%	93,27%	99,53%	96,47%	96,80%
	CoT	**99,52%**	**98,60%**	**99,37%**	**99,73%**	**99,47%**
GPT-4.1	restriktiv	96,98%	92,57%	97,13%	97,87%	95,93%
	einfach	98,02%	94,63%	98,20%	99,20%	96,67%
	CoT	**98,74%**	**96,63%**	**98,00%**	**99,93%**	**98,30%**
Gemma 3 27B	restriktiv	96,51%	91,37%	97,27%	97,37%	94,90%
	einfach	95,62%	88,97%	97,03%	95,00%	94,83%
	CoT	**98,07%**	**94,87%**	**98,40%**	**99,57%**	**96,23%**

Anhang 3.5: Ergebnisübersicht der Few-Shot-Optimierungsstrategien

Modell	**Strategie**	***Overall Acc***	***Document Acc***	***Number Acc***	***Date Acc***	***Amount Acc***
Claude 3 Sonnet	Classic	99,52%	98,57%	99,57%	99,70%	99,30%
	CoT	**99,70%**	**99,10%**	**99,73%**	**99,63%**	**99,73%**
GPT-4.1	Classic	99,22%	97,77%	99,13%	99,83%	98,70%
	CoT	**99,24%**	**97,87%**	**98,83%**	**98,90%**	**99,00%**
Gemma 3 27B	**Classic**	**98,55%**	**95,73%**	**99,07%**	**99,67%**	**96,90%**
	CoT	98,23%	95,10%	98,63%	99,57%	96,50%

Anhang 3.6: Ergebnisübersicht der Evaluationsphase

Modell	**Strategie**	***Overall Acc***	***Document Acc***	***Number Acc***	***Date Acc***	***Amount Acc***
Claude 3 Sonnet	Few-Shot	99,71%	99,21%	99,53%	99,79%	99,82%
	Few-Shot CoT	**99,76%**	**99,36%**	**99,59%**	**99,83%**	**99,87%**
	Zero-Shot CoT	99,67%	99,11%	99,36%	99,82%	99,84%
	einfach	98,62%	95,97%	99,37%	97,24%	99,26%
	Baseline-Prompt	97,60%	96,40%	97,66%	98,04%	97,11%
GPT-4.1	**Few-Shot**	**99,64%**	**99,02%**	**99,49%**	**99,67%**	**99,76%**
	Few-Shot CoT	99,38%	98,34%	99,09%	99,66%	99,40%
	Zero-Shot CoT	99,06%	97,42%	98,16%	99,66%	99,37%
	einfach	99,44%	98,46%	99,33%	99,68%	99,32%
	Baseline-Prompt	98,38%	96,85%	98,21%	98,83%	98,10%
Gemma 3 27B-IT	**Few-Shot**	**99,13%**	**97,61%**	**98,84%**	**99,68%**	**98,87%**
	Few-Shot CoT	98,58%	95,96%	98,66%	99,65%	97,43%
	Zero-Shot CoT	98,68%	96,33%	98,80%	99,52%	97,72%
	einfach	98,00%	94,28%	98,64%	96,64%	98,72%
	Baseline-Prompt	96,74%	94,93%	96,17%	97,34%	96,71%
Gini		**95,53%**	**87,24%**	**96,72%**	**93,37%**	**96,51%**

Anhang 3.7: Vergleich der Modellleistung bei identischer Strategie mit pdfplumber- und Tesseract-Extraktion

Model	***Overall Acc***	***Document Acc***	***Number Acc***	***Date Acc***	***Amount Acc***
Few-Shot-CoT Claude 3 Sonnet PDFPlumber	99,77%	99,37%	99,59%	99,84%	99,87%
Few-Shot-CoT Claude 3 Sonnet Teseract	97,87%	94,43%	95,62%	98,98%	99,01%

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Erkner, 18.09.2025
Ort, Datum

Unterschrift