

PORSCHE

Bachelorarbeit
Studiengang : Data Science

Einsatz von Large Language Models (LLM) zur Extraktion und Strukturierung von Zolldokumenten: Ein KI-gestützter Ansatz zur automatisierten Datenverarbeitung

bei

Porsche AG

von

Kevin Garrison

85826

Betreuender Professor: Prof. Dr. Winfried Bantel
Zweitprüfer : Prof. Dr. Tim Dahmen

Einreichungsdatum : 14. August 2025

Angaben zur Firma

Unternehmen : Porsche AG
Branche : Automobilbranche
Abteilung : Finanzstrategie & Data Science
Adresse : Porscheplatz 1
D - 70435 Stuttgart

Betreuerin : Maike Klepsch (FOD)
Telefon : (+49) 0 152 3 911 0075
E-Mail : maike.klepsch@porsche.de

Eidesstattliche Erklärung

Hiermit erkläre ich, **Kevin Garrison**, dass ich die vorliegenden Angaben in dieser Arbeit bei **Porsche AG** wahrheitsgetreu und selbständig verfasst habe.

Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Aachen 13.08.25

Ort, Datum

Kevin Garrison

Unterschrift (Student)

Kurzfassung

Die Beschaffung von Daten sowie die Abbildung komplexer Geschäftsprozesse in Unternehmen sind häufig mit erheblichem Aufwand verbunden. Insbesondere in diesen Bereichen bieten Automatisierungen durch regelbasierte oder Künstliche Intelligenz (KI)-gestützte Systeme einen erheblichen Mehrwert. Gerade Abteilungen, die sich mit der Verzollung internationaler Warenbewegungen befassen, weisen signifikante Prozessaufwände auf und können von solchen Automatisierungen deutlich profitieren. Aufgrund der häufig mangelhaften und unstrukturierten Datengrundlage ist es jedoch eine Herausforderung, manuelle Prozesse systemseitig abzubilden, ohne dabei Einbußen in Bezug auf Qualität, Nachvollziehbarkeit und Konformität der erzeugten Ergebnisse hinzunehmen.

In dieser Arbeit erfolgt die Implementierung eines KI-gestützten Systems zur Extraktion und Strukturierung relevanter Inhalte aus Zolldokumenten. Der Extraktionsprozess basiert auf dem Einsatz von Large Language Models (LLMs) in Kombination mit gezieltem *Prompt Engineering*, um die semantische Interpretation und die kontextuelle Erfassung der Inhalte zu optimieren. Die Qualität der Ergebnisse wird anhand eines Goldstandards für Zolldaten bewertet. Darüber hinaus erfolgt ein systematischer Vergleich mit einem klassischen hybriden Ansatz, der als Referenzsystem dient und auf Optical Character Recognition (OCR)-gestützter Texterkennung sowie einer Named Entity Recognition (NER)-Komponente beruht. Dabei zeigte sich, dass OCR-bedingte Zeichenerkennungsfehler die Erkennungsleistung des klassischen Systems erheblich beeinträchtigen, während der LLM-basierte Ansatz diese Schwächen durch kontextuelle Interpretation teilweise kompensieren kann. Die besten Resultate hinsichtlich der Micro-Metriken lagen bei einem Recall von 0,53 und einem F1-Score von 0,60. Diese Ergebnisse wurden bei exakter Übereinstimmung numerischer Werte und einem Levenshtein-Schwellenwert von über 90% im Zeichenkettenvergleich zu den Testdaten erzielt. Insgesamt zeigen die Ergebnisse, dass der LLM-Ansatz robuster gegenüber fehlerhaften Eingangsdaten ist und ein höheres Maß an Generalisierungsfähigkeit aufweist, zugleich jedoch auch eigene Limitationen in Bezug auf Reproduzierbarkeit und Extraktionsgenauigkeit mit sich bringt, sodass eine vollständige Automatisierung bestehender Prozesse zur Inhaltsextraktion von Zolldokumenten durch das LLM-basierte System derzeit noch nicht den Anforderungen entspricht und somit nicht ohne menschliche Nachkontrolle eingesetzt werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis	vi
Tabellenverzeichnis	vii
Listings	ix
Abkürzungsverzeichnis	x
1. Einleitung	1
1.1. Problemstellung für die automatisierte Inhaltsextraktion von Dokumenten	1
1.2. Ziel der Arbeit	2
1.3. Abgrenzung der Arbeit	3
2. Grundlagen	4
2.1. Informationsextraktion aus Dokumenten	4
2.2. Optical Character Recognition (OCR)	4
2.2.1. Funktionsweise von Optical Character Recognition (OCR) . . .	5
2.2.2. Tesseract Optical Character Recognition (OCR)	7
2.3. Natural Language Processing (NLP)	8
2.3.1. Named Entity Recognition (NER)	9
2.3.2. Arten benannter Entitäten	10
2.3.3. spaCy: Framework für NLP in Python	11
2.4. Large Language Models (LLM)	12
2.4.1. Funktionsweise eines Large Language Models	12
2.4.2. Transformer	13
2.4.3. Multimodales LLM GPT-4o	15
2.4.4. Prompt Engineering	17
2.5. Definition der Zolldaten	22
2.6. Evaluationsmetriken für die Bewertung des Systems	26
3. Lösungsansatz zur Extraktion der Inhalte von Zolldokumenten	30
4. Implementierung	33
5. Evaluierung des Systems	37

6. Zusammenfassung und Ausblick	43
6.1. Erreichte Ergebnisse	43
6.2. Ausblick	43
Literatur	45
A. Anhang A	49
A.1. Implementierung der Extraktionspipeline	49
A.2. Prompt Design für Datenextraktion	58
A.3. Prompt Design für Schemamigration	67
B. Anhang B	78
C. Anhang C	82

Abbildungsverzeichnis

2.1. Beispiel für Named Entity Recognition mit erkannten Entitäten und zugehörigen Entitätstypen	10
2.2. Publikationen zu NER in den letzten 30 Jahren [15]	11
2.3. Scaled Dot-Produkt Attention und Multi-Head Attention [18].	14
2.4. Transformer-Modell mit Self-Attention-Mechanismus, bestehend aus Encoder, Decoder und Multi-Head Attention [18].	16
3.1. Schematische Übersicht der Datenpipeline zur Entitätsextraktion aus Zolldokumenten. Die Abbildung illustriert die zentralen Verarbeitungsschritte durch ein multimodales LLM	32
5.1. Kopfdaten: Micro-Metriken in Abhängigkeit der Levenshtein-Distanz	42
5.2. Einzelpositionen: Micro-Metriken in Abhängigkeit der Levenshtein-Distanz	42
C.1. Screenshot des LLM basierten Systems zur Zolldokumentenextraktion	82
C.2. Screenshot des hybriden klassischen Systems zur Zolldokumentenextraktion	83

Tabellenverzeichnis

2.1. Prompt mit geringer Temperatur von 0.1 und niedrigem Token-Limit von 5: Die Ausgabe ist sehr vorhersehbar und endet nach 5 Tokens. . .	18
2.2. Prompt mit hoher Temperatur von 0.9 und Top-P 0.5 mit einem hohen Token-Limit von 20: Die Ausgabe ist kreativer und ausführlicher. . . .	18
2.3. Beispiel eines Zero-Shot Prompts mit Parametereinstellung zur Extraktion von Entitäten aus Zolldokumenten	19
2.4. In-Context Learning (ICL) mit Beispielen zur Extraktion von Entitäten aus Zolldokumenten	20
2.5. One-Shot/Few-Shot Prompting zur Extraktion von Entitäten aus Zolldokumenten	21
2.6. instruktionsbasiertes Prompting mit Schrittweisen Anweisungen für Extraktion von Entitäten aus Zolldokumenten	22
2.7. Zentrale Zolldokumente mit Felder für die zu extrahierenden Entitäten der Kopfdaten einer Sendung und Grad des Layouts	23
2.8. Beschreibung der Felder der Kopfdaten	24
2.9. Felder der Einzelpositionen für die Extraktion aus Zolldokumenten . .	25
5.1. Auswertung der aggregierten Metriken - absteigend sortiert nach Recall - für alle Dokumente Promptmethode für die Extraktion durch Gpt-4o in Teilschritt eins	38
5.2. Auswertung der aggregierten Metriken - absteigend sortiert nach Recall - für alle Dokumente pro Promptmethode für die Schemamigration der in Schritt eins gefundenen Entitäten durch Gpt-4o in Teilschritt zwei	39
5.3. Auswertung spaCy NER für die Extraktion Teilschritt eins	39
5.4. Micro-basierte Metriken zur Bewertung der Kopfdatenextraktion . . .	40
5.5. Macro-basierte Metriken zur Bewertung der Kopfdatenextraktion . .	40
5.6. Micro-basierte Metriken zur Bewertung der Extraktion der Einzelpositionen	41
5.7. Macro-basierte Metriken zur Bewertung der Extraktion der Einzelpositionen	41
B.1. Klassenspezifische Extraktionsmetriken der Kopfdaten für 30 Sendungen mit der Methode Zero-Shot Prompting (erster Teil).	79
B.2. Klassenspezifische Extraktionsmetriken der Kopfdaten für 30 Sendungen mit der Methode Zero-Shot Prompting (zweiter Teil).	80

B.3. Klassenspezifische Extraktionsmetriken der Einzelpositionen für 30 Sendungen mit der Methode Zero-Shot Prompting.	81
---	----

Listings

4.1. LLM als OCR-Ersatz – PDF zu Bild und Bild zu Text Extraktion . . .	33
4.2. Schemamigration des extrahierten JSON-Contents aus Teilaufgabe eins	34
4.3. Pydantic Modell für eine Schemaüberprüfung	35
A.1. LLM als OCR Ersatz - Umwandlung von PDF in Bild und Extraktion der Entitäten von Bild zu Text	49
A.2. LLM um Daten in vordefiniertes Schema zu migrieren	54

Abkürzungsverzeichnis

CCA Connected Component Analysis. 6, 7

CNN Convolutional Neural Network. 6

GPT-4o Generative Pre-trained Transformer 4 Omni. 3, 15

IBL Instruction-Based Learning. 35

ICL In Context Learning. vii, 2, 19, 20

KI Künstliche Intelligenz. iii, 9

LLM Large Language Model. iii, iv, vi, 2, 3, 12, 13, 15, 17, 19–22, 26, 30–35, 37, 39, 40, 44

LSTM Long Short-Memory Network. 6

MLLM Multimodales Large Language Model. 31

NER Named Entity Recognition. iii, vi, vii, 2, 3, 8–11, 15, 30, 31, 33, 39

NLG Natural Language Generation. 8, 15

NLP Natural Language Processing. 8, 9, 11–13, 15, 30

NLU Natural Language Understanding. 8, 15

OCR Optical Character Recognition. iii, 2–8, 15, 30, 31, 33, 39

RLHF Reinforcement Learning from Human Feedback. 30

1. Einleitung

1.1. Problemstellung für die automatisierte Inhaltsextraktion von Dokumenten

Die in Unternehmen anfallenden Datenmengen unterliegen einem stetigen, teils exponentiellen Wachstum. Dies betrifft sämtliche Phasen der datenbezogenen Wertschöpfung – von der Erfassung und Vorverarbeitung bis hin zur modellgestützten Abbildung geschäftsrelevanter Prozesse innerhalb der Hauptverarbeitung. Vor diesem Hintergrund stoßen manuelle Verfahren zur Verarbeitung und Analyse dieser Daten zunehmend an ihre Grenzen – insbesondere im Hinblick auf Effizienz, Skalierbarkeit und Fehleranfälligkeit.

Besonders betroffen sind Unternehmensbereiche, die mit der internationalen Verzollung von Waren befasst sind, da hierbei eine Vielzahl komplexer und unstrukturierter Zolldokumente verarbeitet werden muss. Die Automatisierung der Inhaltsextraktion von Dokumenten aus dem Zollkontext stellt daher eine zentrale Herausforderung dar, um den gestiegenen Anforderungen an Geschwindigkeit, Genauigkeit und Nachvollziehbarkeit in der Abwicklung gerecht zu werden.

Zolldokumente liegen häufig als zusammengeführte PDF-Dateien vor, unterscheiden sich jedoch erheblich in Layout, Struktur und Qualität (vgl. Abschnitt 2.5). Zudem enthalten sie oftmals keinen eingebetteten digitalen Text, sondern bestehen lediglich aus gescannten oder fotografierten Bildern. Diese Vielfalt macht sowohl die manuelle als auch die automatisierte Extraktion der für die Zollabwicklung relevanten Inhalte zu einem aufwändigen und fehleranfälligen Prozess. Dabei existieren einerseits standardisierte Formate wie das Zolldokument T1, die Warenverkehrsbescheinigung oder das Präferenzdokument. Andererseits gibt es zahlreiche nicht standardisierte Dokumente, deren Gestaltung und Struktur je nach Lieferant stark variieren.

Klassische Verfahren zur Informationsextraktion aus Dokumenten basieren häufig auf regelbasierten Ansätzen, heuristischen Mustern oder vordefinierten Templates. Diese Methoden stoßen jedoch bei unstrukturierten oder variantenreichen Dokumenten schnell an ihre Grenzen. Sie erfordern umfangreiche manuelle Konfigurationen, sind wenig skalierbar und reagieren äußerst sensitiv auf Layout- oder Sprachänderungen [1]. Schon geringfügige Abweichungen, etwa in der Positionierung von Elementen oder der Formulierung relevanter Informationen, führen häufig zu fehlerhaften oder fehlenden Extraktionen. Zudem fehlt es regelbasierten Systemen oft an kontextuellem Verständnis, was die semantische Interpretation der Inhalte

erschwert – insbesondere bei mehrdeutigen Begriffen oder domänenspezifischen Begriffen .

Moderne Verfahren auf Basis großer Sprachmodelle (Large Language Models (LLMs)) versprechen hier eine deutliche Verbesserung. Aufgrund ihrer enormen Trainingsdatenbasis und generativen Fähigkeiten können sie flexibel auf unterschiedliche Layouts, Sprachvarianten und Kontexte reagieren. Insbesondere durch In Context Learning (ICL) (vgl. Tabelle 2.4) und das Einbinden geeigneter Prompts lassen sich LLMs dazu befähigen, strukturierte Informationen auch aus heterogenen Dokumenten zu extrahieren. Dennoch bestehen auch hier Herausforderungen: Die Modelle erfordern geeignete Kontextrepräsentationen und ihre Vorhersagen sind weniger deterministisch kontrollierbar, da sie auf probabilistischen Wahrscheinlichkeitsverteilungen über ein Vokabular basieren [2]. Zudem kann es zu Halluzinationen kommen, bei denen Modelle nicht vorhandene Informationen generieren oder Fakten kontextuell falsch zuordnen. Der zuverlässige Einsatz in hochregulierten Bereichen wie der Zollabwicklung erfordert daher zusätzliche Absicherungsmechanismen sowie eine gezielte Modellsteuerung, etwa durch prompt engineering, strukturierte Ausgaben oder nachgelagerte Validierungslogik [3].

1.2. Ziel der Arbeit

Diese Arbeit untersucht, inwieweit ein automatisiertes System mithilfe eines LLMs die bestehenden Herausforderungen bei der Extraktion von Zolldaten bewältigen kann und welche neuen Schwächen und Hürden dabei möglicherweise entstehen. Angesichts der Vielzahl an Dokumentenarten und der hohen Layout-Varianz stoßen regelbasierte Systeme ohne gezielte Feinjustierung auf die jeweiligen Anwendungsfälle schnell an ihre Grenzen. Zwar sind klassische, regelbasierte Verfahren in kontrollierten Anwendungsfällen robust und effizient, zeigen jedoch eine hohe Fehleranfälligkeit bei struktureller Varianz in Dokumentenlayouts und können die Heterogenität domänenspezifischer Inhalte nur unzureichend abbilden [4].

Moderne Ansätze, insbesondere LLMs, wurden auf sehr großen und diversifizierten Datensätzen trainiert und zeigen dadurch eine hohe Generalisierungsfähigkeit. Dennoch offenbaren sie häufig Schwächen bei der Extraktion domänenspezifischer Informationen [3].

Zur Untersuchung der jeweiligen Stärken und Schwächen erfolgt in dieser Arbeit eine vergleichende Gegenüberstellung eines hybriden, klassischen Verfahrens – bestehend aus einer vorgelagerten optischen Zeichenerkennung (Optical Character Recognition (OCR)) (vgl. Abschnitt 2.2) und einer nachgelagerten Entitätserkennung mittels der benannten Entitätserkennung (Named Entity Recognition (NER)) (vgl. Unterabschnitt 2.3.1) – mit einem modernen Ansatz, der beide Teilschritte durch ein LLM-basiertes System abbildet.

1.3. Abgrenzung der Arbeit

Im Folgenden werden die inhaltlichen und methodischen Grenzen dieser Arbeit dargelegt.

Zunächst liegt der Schwerpunkt der Arbeit auf der Extraktion und Strukturierung relevanter Inhalte aus Zolldokumenten mithilfe eines Large Language Models. Dabei wird auf eine bestehende Optical Character Recognition-Komponente (*Tesseract*¹ vgl. Unterabschnitt 2.2.2) sowie auf ein vortrainiertes Named Entity Recognition-Modell von *spaCy*² (vgl. Unterabschnitt 2.3.3) zurückgegriffen, ohne diese im Detail zu optimieren. Dieser hybride Ansatz dient lediglich als Vergleichsbasis für das zu entwickelnde System.

Des Weiteren ist die Prüfung der rechtlichen oder formalen Gültigkeit von Zolldokumenten explizit nicht Gegenstand dieser Arbeit. Ziel ist ausschließlich die technische Extraktion inhaltlich relevanter Informationen, unabhängig von rechtlichen Anforderungen.

Zur Eingrenzung des Umfangs beschränkt sich die Arbeit auf eine Auswahl spezifischer Dokumenttypen (vgl. Abschnitt 2.5). Eine vollständige Abdeckung aller zollrelevanten Formulare im internationalen Handel wird nicht angestrebt.

Die Evaluation der entwickelten Extraktionspipeline basiert auf einem begrenzten Datensatz realer Zolldokumente. Aufgrund der beschränkten Datenbasis von 30 Datenpunkten, kann keine Aussage über die Generalisierbarkeit der Ergebnisse auf alle möglichen Layouts und Varianten von Zolldokumenten getroffen werden.

Darüber hinaus wird in dieser Arbeit ausschließlich ein spezifisches Large Language Model – Generative Pre-trained Transformer 4 Omni (GPT-4o) (vgl. Unterabschnitt 2.4.3) – eingesetzt. Ein Vergleich mit anderen Modellen erfolgt nicht. Ziel ist es nicht, einen umfassenden Vergleich diverser Modelle durchzuführen, sondern das Potenzial eines exemplarischen Modells zur Extraktion aus unstrukturierten Zolldaten zu untersuchen.

¹<https://github.com/tesseract-ocr/tesseract>

²<https://spacy.io/api>

2. Grundlagen

2.1. Informationsextraktion aus Dokumenten

In einem Zeitalter von Big Data stellt die Extraktion von wertvollem Wissen aus komplexen, unstrukturierten Daten eine immer größere Hürde dar. Automatisierte Informationsextraktion aus Dokumenten bildet hierbei einen wesentlichen Bestandteil moderner Datenverarbeitung. Diese umfasst mehrere essenzielle Schritte, die für eine präzise und verlustfreie Erfassung relevanter Inhalte berücksichtigt werden müssen. Eine zentrale Voraussetzung ist die Kenntnis über das Dokumentenlayout sowie die logische Struktur des jeweiligen Dokuments, da andernfalls kontextabhängige Informationen verloren gehen können [5]. Ein typisches Dokument besteht aus verschiedenen strukturellen Elementen wie Fließtexten, Tabellen oder Abbildungen. Diese Elemente werden im Rahmen einer Layoutanalyse (vgl. Unterabschnitt 2.2.1) identifiziert und klassifiziert, sodass sie im weiteren Verlauf der Verarbeitung als kontextuelle Bezugspunkte dienen können [6]. Handelt es sich um gescannte oder handgeschriebene Dokumente, müssen zunächst die enthaltenen Zeichen und Wörter erkannt und in digitale Textrepräsentationen überführt werden. Diese Aufgabe erfordert eine präzise Zeichen- und Zeilensegmentierung und gegebenenfalls eine Schräglagenkorrektur des Dokuments. In der Praxis wird dieser Verarbeitungsschritt meist durch spezialisierte Optical Character Recognition (OCR)-Systeme (vgl. Abschnitt 2.2) umgesetzt, welche die visuelle Information in maschinenlesbaren Text umwandeln [7].

2.2. Optical Character Recognition (OCR)

Die optische Zeichenerkennung (Optical Character Recognition (OCR)) bezeichnet ein Verfahren zur automatisierten Umwandlung von handschriftlichem oder gedrucktem Text in gescannten Dokumenten oder Bildern in digitalen Text. Dabei werden die einzelnen Zeichen erkannt und umgewandelt, wobei regelbasierte, statistische oder Deep-Learning-Verfahren zum Einsatz kommen.

Die ersten Verfahren zur optischen Zeichenerkennung entstanden in den 1950er-Jahren. Diese waren jedoch in ihrer Leistung stark eingeschränkt, konnten nur eine geringe Anzahl an Zeichen verarbeiten und arbeiteten zudem langsam und unpräzise [4]. Moderne Verfahren basieren auf Deep Learning und sind dadurch

dynamischer und leistungsfähiger [7]. Moderne OCR-Systeme bieten sogenannte Ende-zu-Ende-Lösungen, bei denen sämtliche Verarbeitungsschritte – von der Vorverarbeitung über die Zeichenerkennung bis hin zur Nachverarbeitung – vollständig integriert und aufeinander abgestimmt sind. Damit entfällt die Notwendigkeit externer Module für Aufgaben wie Binarisierung, Rauschunterdrückung oder Schräglagenkorrektur (vgl. Unterabschnitt 2.2.1). Dies erleichtert die Anwendung und erhöht die Genauigkeit sowie Zuverlässigkeit der Texterkennung in verschiedensten Einsatzszenarien.

2.2.1. Funktionsweise von Optical Character Recognition (OCR)

Ein OCR-System durchläuft dabei mehrere technische Verarbeitungsstufen. Nach der Erfassung des Dokuments durch Scannen oder Fotografie erfolgt zunächst die Vorverarbeitung, bei der Maßnahmen wie Binarisierung, Rauschunterdrückung und Schräglagenkorrektur angewendet werden, um die Bildqualität und den Informationsgehalt insgesamt zu verbessern. Die Binarisierung überführt die Bilddaten in Schwarz-Weiß-Pixel, wodurch die Trennung von Zeichen und Hintergrund anhand eines definierten Schwellenwerts erleichtert wird und eine effizientere Segmentierung möglich ist. Dieser Schritt ist besonders wichtig, da viele nachfolgende Algorithmen – etwa zur Zeichenerkennung oder Segmentierung – davon ausgehen, dass der Text deutlich vom Hintergrund unterscheidbar ist. In der Praxis wird hierzu meist ein globaler oder adaptiver Schwellenwert verwendet. Beim globalen Ansatz wird für das gesamte Bild ein einziger Schwellenwert festgelegt, während beim adaptiven Verfahren für kleine Bildbereiche jeweils eigene Schwellenwerte berechnet werden. Dadurch können auch Dokumente mit ungleichmäßig ausgeleuchtetem Hintergrund zuverlässig in klare Schwarz-Weiß-Bilder umgewandelt werden, was die Erkennungsgenauigkeit erhöht [8].

Bei der Rauschunterdrückung werden unerwünschte Bildstörungen wie zufällige Punkte, Flecken oder kleine Linien entfernt, die das Erkennen von Zeichen erschweren könnten. Solche Störungen entstehen häufig durch Staub auf dem Scanner, Papierstruktur oder elektronische Bildartefakte. Die Rauschunterdrückung kann beispielsweise mit Hilfe der Faltung (Konvolution) realisiert werden, bei der ein sogenannter Glättungsfilter, wie etwa ein gleitender Mittelwerts-Filter, über das Bild gelegt wird. Dieser Filter berechnet für jeden Pixel den Durchschnitt der benachbarten Pixelwerte und ersetzt den ursprünglichen Wert durch diesen Durchschnitt. Dadurch werden plötzliche Helligkeitsänderungen und kleine Störungen geglättet, während die grundlegenden Strukturen des Textes erhalten bleiben. Je nach Art und Ausmaß des Rauschens können unterschiedliche Filtertypen und -größen zum Einsatz kommen, um ein möglichst optimales Ergebnis für die weitere Verarbeitung zu erzielen. Die Schräglagenkorrektur sorgt dafür, dass schief gescannte oder fotografierte Dokumente ausgerichtet werden, sodass die Textzeilen wieder horizontal verlaufen und eine bessere Weiterverarbeitung gewährleistet

ist. Bereits geringe Schräglagen können die automatische Texterkennung und die anschließende Segmentierung deutlich erschweren, da viele Algorithmen davon ausgehen, dass der Text gerade angeordnet ist. In der Praxis wird die Schräglage häufig durch Analyse der Verteilung der schwarzen Pixel in den einzelnen Bildzeilen ermittelt: Dazu werden verschiedene kleine Drehwinkel ausprobiert und jeweils überprüft, bei welchem Winkel die Zeilen im Bild am besten ausgeprägt erscheinen, also der Kontrast zwischen Textzeilen und Zwischenräumen am größten ist. Das Bild wird dann entsprechend gedreht, sodass die Textzeilen optimal horizontal ausgerichtet sind. Durch diese Korrektur wird die Grundlage für eine fehlerarme Texterkennung und weiterführende Verarbeitung geschaffen und es erfolgt die Segmentierung, bei der das Bild in Einheiten wie Zeichen oder Wörter unterteilt wird. Dies kann entweder explizit durch geometrische Trennverfahren oder implizit durch lernbasierte Ansätze erfolgen. Zu den einfacheren expliziten Verfahren zählen die Verbundene-Komponenten-Analyse (Connected Component Analysis (CCA)) sowie die Projektionsprofil-Methode (Projection Profiles) (vgl. Unterabschnitt 2.2.1). Die CCA-Methode identifiziert zusammenhängende Pixelbereiche im Bild – sogenannte zusammenhängende Komponenten – wobei jede Komponente als potenzielles Zeichen betrachtet wird. Die Projektionsprofil-Methode nutzt vertikale Projektionen zur Trennung einzelner Zeichen innerhalb einer Zeile und horizontale Projektionen zur Abgrenzung der Textzeilen. In komplexeren Dokumenten, etwa bei überlappenden Zeichen oder starkem Rauschen, stoßen diese einfachen Verfahren jedoch an ihre Grenzen, sodass auf fortgeschrittene Segmentierungsmethoden, wie zum Beispiel Clustering-Algorithmen, zurückgegriffen werden muss [4].

Bevor die eigentliche Klassifikation der Zeichen erfolgt, wird zunächst eine Merkmalsextraktion durchgeführt. Während regelbasierte und statistische OCR-Systeme explizit definierte Merkmale aus den Zeichenbildern extrahieren, übernehmen moderne Deep-Learning-basierte Verfahren diese Aufgabe implizit: Die neuronalen Netze lernen während des Trainingsprozesses automatisch, relevante Merkmale direkt aus den Rohdaten zu identifizieren. Anschließend werden dann die erkannten Zeichen mittels eines Klassifikationsmodells einer bestimmten Klasse zugeordnet. Je nach System kommen hierbei einfache Heuristiken, statistische Modelle oder Deep-Learning-Architekturen wie Convolutional Neural Network (CNN) oder Long Short-Memory Network (LSTM) zum Einsatz [9]. Die Erkennungsgenauigkeit kann nach der Klassifikation durch verschiedene Strategien weiter gesteigert werden. Dazu zählen etwa die Kombination mehrerer Klassifikatoren in sequentieller, paralleler oder hierarchischer Struktur sowie die kontextbasierte Analyse angrenzender Zeichen oder Wörter, um zusätzliche semantische Informationen zur Validierung heranzuziehen [4].

2.2.2. Tesseract Optical Character Recognition (OCR)

Ein weit verbreitetes OCR-System ist Tesseract, welches ursprünglich zwischen 1984 und 1994 von Hewlett-Packard (HP) entwickelt wurde und seit der Veröffentlichung als Open Source im Jahr 2006-2018 von Google und mittlerweile von der Community aktiv weiterentwickelt wird. Ab Version 4 basiert Tesseract auf einem hybriden Ansatz, der klassische bildanalytische Verfahren mit Deep-Learning-Methoden kombiniert. Insbesondere kommt ab dieser Version ein rekurrentes neuronales Netz auf Basis von LSTM zum Einsatz, das nicht mehr einzelne Zeichen isoliert klassifiziert, sondern ganze Textzeilen sequenziell verarbeitet. Dadurch kann das Modell den Kontext benachbarter Zeichen nutzen, was zu einer deutlich robusteren und genaueren Zeichenerkennung führt.

Die Verarbeitungsarchitektur von Tesseract umfasst mehrere aufeinanderfolgende Verarbeitungsstufen. Zu Beginn erfolgt die Vorverarbeitung, wobei Tesseract ein binarisiertes Eingabebild erwartet. Anschließend übernimmt eine interne Layoutanalyse die Identifikation von Textregionen, Zeilen und Wörtern. Mittels der CCA werden zusammenhängende Pixelbereiche – sogenannte Blobs – identifiziert, die potenziell einzelnen Zeichen oder Zeichenkomponenten entsprechen.

Im nächsten Schritt werden diese Blobs zu Textzeilen und Wörtern segmentiert. Die Zeilenfindung basiert auf einer Analyse der vertikalen Struktur der Blobs und kann auch schräg gescannte Seiten bis zu einem gewissen Grad ohne explizite Schräglagenkorrektur verarbeiten. Je nach Schriftart unterscheidet Tesseract zwischen fester und proportionaler Zeichenbreite, was Einfluss auf die Segmentierung von Wörtern und Zeichen hat. Die Zeichenerkennung erfolgt im Standardmodell ab Version 4 ausschließlich durch das LSTM-basierte neuronale Netzwerk, das die gesamte Texterkennung als sequenziellen End-to-End-Prozess durchführt. Die ältere, zweiphasige Klassifikation mit statischem und adaptivem Klassifikator steht noch als Legacy-Modus zur Verfügung, wird aber in modernen Anwendungsfällen nicht mehr verwendet.

Ergänzend zur eigentlichen Zeichenerkennung wird eine linguistische Nachverarbeitung durchgeführt. Tesseract bewertet dabei unterschiedliche Wortkandidaten anhand von Heuristiken wie Groß- und Kleinschreibung, numerischer Struktur, Wörterbuchabgleich oder Worthäufigkeit. Obwohl kein vollständiges probabilistisches Sprachmodell integriert ist, nutzt das System Distanzwerte und Ranglisten zur Auswahl der plausibelsten Wortform.

Trotz eines vergleichsweise kleinen Trainingsdatensatzes von lediglich 60.160 Instanzen erzielt Tesseract eine hohe Erkennungsgenauigkeit. Andere publizierte Klassifikatoren, wie etwa die Systeme von Calera oder Baird, wurden hingegen auf über einer Million Datenpunkten trainiert.

Die Kombination aus klassischer Bildverarbeitung, leistungsstarker LSTM-basierter Erkennungs-Engine und sprachlicher Heuristik macht diese Engine besonders leistungsfähig bei maschinengedruckten Texten. In komplexeren Anwendungsfällen, wie etwa bei stark strukturierten oder verrauschten Dokumenten, stößt das System

jedoch an seine Grenzen, was Potenzial für die Integration externer Sprachmodelle oder spezialisierter Vorverarbeitungstechniken bietet [10].

2.3. Natural Language Processing (NLP)

Die natürliche Sprachverarbeitung (Natural Language Processing (NLP)) ist ein bedeutender Teilbereich sowohl der Künstlichen Intelligenz als auch der Linguistik. Sie ermöglicht es, automatisiert Informationen aus unstrukturierten, textbasierten Daten zu extrahieren. Erste Ansätze zur Sprachverarbeitung lassen sich bis in die 1950er-Jahre zurückverfolgen. Damals war sie noch deutlich von der klassischen Textinformationssuche abgegrenzt; in der heutigen Praxis sind beide Bereiche jedoch zunehmend miteinander verschmolzen [11].

NLP lässt sich in zwei grundlegende Teilgebiete untergliedern: das Sprachverständnis (Natural Language Understanding (NLU)) und die Sprachgenerierung (Natural Language Generation (NLG)). Während NLU auf die maschinelle Analyse von Satzstruktur, Bedeutung und Kontext abzielt, beschäftigt sich NLG mit der automatisierten Erzeugung von Text, etwa im Rahmen von Textzusammenfassungen oder Textgenerierung.

Ziel der natürlichen Sprachverarbeitung ist es, spezifische Aufgaben innerhalb algorithmischer Systeme zu übernehmen. Zahlreiche Teilbereiche finden bereits Anwendung in der Praxis – beispielsweise die automatische Textzusammenfassung, die maschinelle Übersetzung oder die Erkennung benannter Entitäten (NER), etwa von Organisationen oder Personennamen in Texten. Auch die optische Zeichenerkennung (OCR) zählt hierzu, mit der Texte auf Bildern oder in gescannten Dokumenten digitalisiert werden können.

Basierend auf diesen Anwendungsfeldern lassen sich modulare Verarbeitungssysteme nach dem Prinzip der Unix-Pipeline konzipieren. Dabei wird die Ausgabe eines Subsystems als Eingabe für das nächste genutzt, wodurch sich komplexe Sprachverarbeitungssysteme aufbauen lassen. Diese Systeme können je nach Anwendungsfall sequentiell, parallel oder hierarchisch organisiert sein [11] und ermöglichen eine flexible, austauschbare Architektur [12].

Bevor es zur Sprachverarbeitung kommt, muss der Text zuerst vorverarbeitet werden. Ohne die Vorverarbeitung sind NLP-Prozesse gar nicht möglich.

Im ersten Schritt kann der Text optional normalisiert werden, bedeutet in eine standardisierte Form gebracht werden. Dies erreicht man durch Umwandeln aller Buchstaben in Kleinbuchstaben, Entfernen von Sonderzeichen und Entfernen oder Umwandeln von Umlauten und Akzenten.

Dann folgt die Tokenisierung, das ist die Zerlegung eines Textes in kleinere Einheiten, sogenannte Tokens [13]. Diese Tokens können Wörter, Satzzeichen oder Untereinheiten von Wörtern sein und bilden eine zentrale Grundlage für die algorithmische Verarbeitung natürlicher Sprache. Nach der Tokenisierung werden die Tokens anhand eines festgelegten Vokabulars einer ID zugeordnet und anschließend

jeder Token in einen hochdimensionalen Vektor transformiert, sogenannte Input Embeddings, die semantische Informationen über die Bedeutung der einzelnen Tokens enthalten. Moderne Transformer-Modelle ergänzen diese Embeddings zusätzlich um Positional Encodings, welche Informationen zur Reihenfolge der Tokens in der Eingabesequenz bereitstellen. Diese mehrstufige Aufbereitung, bestehend aus Normalisierung, Tokenisierung, Einbettung und Positionskodierung, bereitet die Eingabedaten auf, sodass mit NLP-Methoden syntaktische und semantische Zusammenhänge erkannt und weiterverarbeitet werden können [14].

2.3.1. Named Entity Recognition (NER)

Destillierte Informationsextraktion aus unstrukturierten Daten wie z.B. Dokumenten ist ein sehr wichtiges Teilgebiet der natürlichen Sprachverarbeitung und ist unerlässlich, um Künstliche Intelligenz (KI)-Systeme für reale Anwendungen zu implementieren. Die Named Entity Recognition (NER) ist eine Methode der natürlichen Sprachverarbeitung, die dazu dient, Entitäten aus der realen Welt automatisiert zu erkennen und sie vordefinierten semantischen Kategorien (z.B. Personen, Organisationen oder Orte) zuzuordnen. Die Umsetzung erfolgt entweder regelbasiert oder unter Einsatz von Verfahren des maschinellen Lernens [15].

Abbildung 2.2 zeigt deutlich den Anstieg der Publikationen im Bereich der NER in den letzten 30 Jahren – insbesondere bedingt durch neue Modellarchitekturen. Eine zentrale Rolle spielt hierbei auch die Transformer-Architektur (vgl. Unterabschnitt 2.4.2).

Die NER ermöglicht es, relevante Entitäten aus unstrukturiertem Text zu identifizieren und herauszufiltern, um diese im nächsten Schritt in strukturierte Daten umzuwandeln oder für weitere Schritte eines übergeordneten Systems zur Verfügung zu stellen. Dies können unter anderem Datum, Namen, Events, Orte oder auch numerische Werte sein [12].

Sei T eine gegebene Token-Sequenz (vgl. Abschnitt 2.3)

$$\{x_0, x_1, \dots, x_{m-1}\},$$

deren Ziel es ist, eine Menge vordefinierter Entitäten zu identifizieren. Dabei besteht die Aufgabe darin, eine Menge von Tupeln

$$(I_s, I_e, l) \quad \text{mit} \quad I_s, I_e \in [1, N], I_s \leq I_e, l \in \mathcal{L}$$

zu erzeugen. Hierbei bezeichnen I_s und I_e den Beginn bzw. das Ende einer Entität in der Sequenz, während l die zugehörige Kategorie aus einer vordefinierten Menge \mathcal{L} repräsentiert.

Ein typisches Beispiel – dargestellt in Abbildung 2.1 – ist der Satz: „Barack Obama wurde in Honolulu geboren.“ Ein NER-System erkennt darin „Barack Obama“ als

Entität des Typs *Person* und „Honolulu“ als Entität des Typs *Ort* (vgl. [12]).

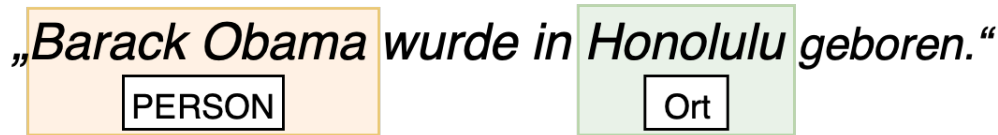


Abbildung 2.1.: Beispiel für Named Entity Recognition mit erkannten Entitäten und zugehörigen Entitätstypen

2.3.2. Arten benannter Entitäten

Im Rahmen der NER lassen sich benannte Entitäten anhand ihrer strukturellen Eigenschaften in unterschiedliche Kategorien einteilen: verschachtelte, nicht-fortgesetzte und fortgesetzte Entitäten [12].

Verschachtelte benannte Entitäten

Verschachtelte benannte Entitäten (nested named entities) treten auf, wenn eine Entität vollständig oder teilweise in einer anderen enthalten ist. Ein Beispiel ist der Satz: „Barack Obama wurde in Honolulu, Hawaii geboren.“ Hier stellt „Honolulu, Hawaii“ eine verschachtelte Entität dar, wobei „Honolulu“ als Hauptstadt und „Hawaii“ als US-Bundesstaat jeweils eigenständige geografische Entitäten bilden.

Nicht-fortgesetzte benannte Entitäten

Nicht-fortgesetzte benannte Entitäten (non-continued named entities) bestehen aus zusammenhängenden, klar abgegrenzten Textabschnitten. Im Satz „Google wurde von Larry Page und Sergey Brin gegründet“ lassen sich die Entitäten „Google“, „Larry Page“ und „Sergey Brin“ eindeutig als voneinander getrennte Einheiten erkennen. Klassische NER-Modelle können diese Form der Entitäten durch einfache Sequenzlabeling-Methoden zuverlässig extrahieren.

Fortgesetzte benannte Entitäten

Zuletzt gibt es noch die fortgesetzten benannten Entitäten (continued named entities), diese setzen sich aus mehreren, nicht direkt aufeinanderfolgenden Textbestandteilen zusammen, die dennoch eine gemeinsame semantische Einheit bilden. Ein Beispiel ist der Satz: „Der Patient zeigte einen produktiven Husten mit weißem

oder blutigem Auswurf.“ Die Wortgruppen „Husten mit weißem Auswurf“ und „Husten mit blutigem Auswurf“ beziehen sich beide auf dasselbe Symptom und bilden zusammen eine fortgesetzte Entität. Die Erkennung solcher Strukturen erfordert kontextsensitives Modellverhalten, wie es etwa durch Transformer-basierte Architekturen ermöglicht wird.

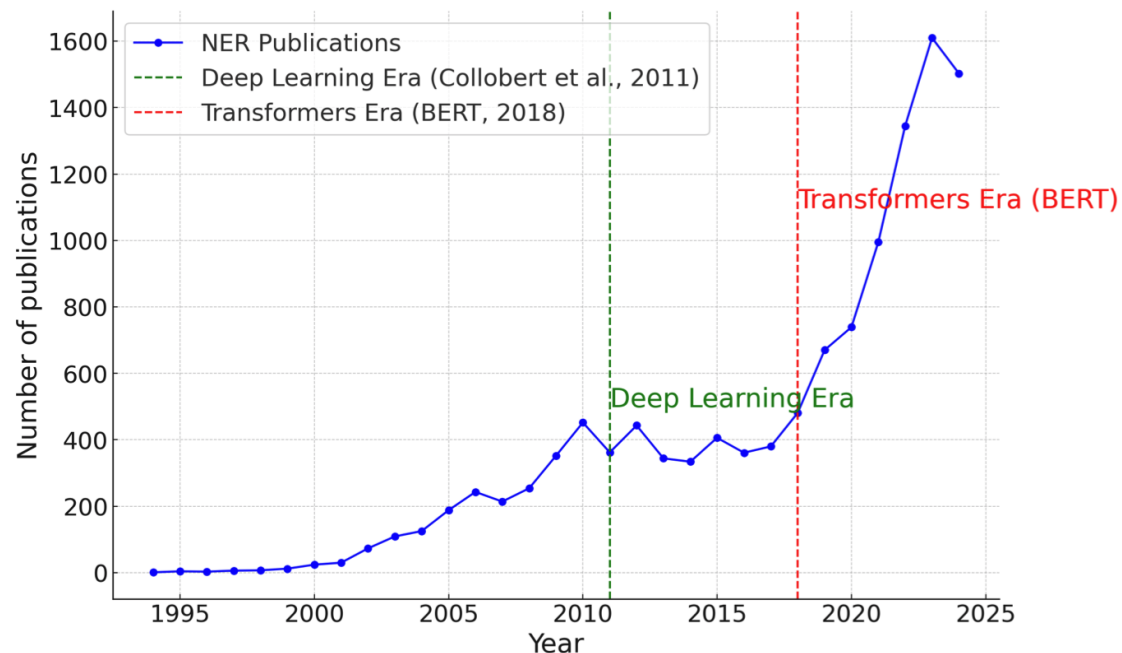


Abbildung 2.2.: Publikationen zu NER in den letzten 30 Jahren [15]

2.3.3. spaCy: Framework für NLP in Python

Ein etabliertes und weit verbreitetes Framework für fortgeschrittene Anwendungen im Bereich des Natural Language Processing (NLP) ist *spaCy*³. Es wurde im Jahr 2015 von der Firma Explosion AI veröffentlicht und ist in Python sowie Cython implementiert.

spaCy ermöglicht die effiziente Entwicklung vollständiger NLP-Pipelines auf Produktionsniveau, ohne das Python-Ökosystem verlassen zu müssen. Die Architektur des Frameworks ist dabei modular aufgebaut und bietet standardisierte Komponenten wie Tokenisierung, Lemmatisierung, Part-of-Speech-Tagging, NER sowie syntaktisches Parsing.

Ein wesentliches Merkmal von spaCy ist seine Orientierung an praktischen, performanten Anwendungen: Statt einer Vielzahl theoretischer Methoden liegt der Fokus auf Geschwindigkeit, Robustheit und einfacher Integration in produktive Systeme.

³<https://spacy.io/api>

Durch eine aktive Entwickler-Community sowie regelmäßige Aktualisierungen fließen aktuelle Forschungsergebnisse kontinuierlich in die Weiterentwicklung ein [13].

2.4. Large Language Models (LLM)

Ein Meilenstein im NLP stellen große Sprachmodelle (Large Language Models (LLMs)) dar, da sie durch ihre besondere Architektur (vgl. Unterabschnitt 2.4.2) in sehr vielen Bereichen einsetzbar sind. Bei einem LLM handelt es sich um ein komplexes mathematisches Modell, das in der Lage ist, auf Basis eines gegebenen Texteingangs eine Wahrscheinlichkeitsverteilung über einen gewissen Wortschatz zu berechnen. Die Eingabe erfolgt in natürlicher Sprache und wird intern zunächst in Texttokens umgewandelt, die in Form von numerischen Vektoren (Embeddings) repräsentiert werden. Diese dienen dem Modell als Grundlage für die Inferenz. Die resultierenden Embeddings werden anschließend wieder in natürliche Sprache dekodiert.

Das Training eines LLM erfolgt auf sehr großen Textdatensätzen aus dem Internet, wobei die Modellparameter (Gewichte) durch Optimierungstechniken angepasst werden [16]. Dadurch ist das Modell in der Lage, komplexe Strukturen der Sprache zu erfassen und kontextabhängige Zusammenhänge zu erkennen, was den Eindruck einer intelligenten Interaktion vermittelt.

Technisch basieren LLMs auf der sogenannten Transformer-Architektur, die als Meilenstein in der Verarbeitung natürlicher Sprache (NLP) gilt. Durch den Einsatz des sogenannten Attention-Mechanismus ist es möglich, den gesamten Textkontext zu berücksichtigen, was gegenüber früheren Modellen wie Recurrent Neural Networks (RNNs) oder Long Short-Term Memory Networks (LSTMs) eine signifikante Verbesserung darstellt. Zudem ermöglicht dieser Mechanismus die parallele Verarbeitung langer Textsequenzen (vgl. Unterabschnitt 2.4.2) [17].

Diese Eigenschaften machen LLMs äußerst flexibel einsetzbar. Typische Anwendungsfelder umfassen unter anderem die Zusammenfassung von Texten, maschinelle Übersetzung, Textgenerierung sowie die gezielte Informationsextraktion aus Dokumenten (vgl. Abschnitt 2.1) [14].

2.4.1. Funktionsweise eines Large Language Models

Ein LLM erhält als Eingabe eine Sequenz von Tokens (vgl. Abschnitt 2.3)

$$\{x_0, x_1, \dots, x_{m-1}\},$$

und generiert für jede Position i eine Wahrscheinlichkeitsverteilung

$$\Pr(\cdot \mid x_0, \dots, x_{i-1})$$

über das Vokabular des Modells. Diese Verteilung beschreibt die Wahrscheinlichkeit, mit der ein mögliches nächstes Token x_i basierend auf dem bisherigen Kontext generiert wird. Das Modell folgt dabei einem autoregressiven Ansatz, wodurch zu jedem Zeitpunkt das nächste Token ausschließlich auf Grundlage der vorangegangenen Tokens vorhergesagt wird. LLM werden als *large* bezeichnet, da sie typischerweise über mehrere Milliarden Parameter verfügen und auf umfangreichen Datensätzen trainiert werden. Ihre Leistungsfähigkeit zeigt sich unter anderem in der Fähigkeit zu generalisieren und eine Vielzahl an Aufgaben der natürlichen Sprachverarbeitung NLP ohne aufgabenspezifisches Fine-Tuning der Modellparameter zu lösen [16].

2.4.2. Transformer

Der Transformer stellt seit seiner Einführung durch Google die grundlegende Architektur für moderne LLMs dar und hat klassische sequenzielle Modelle wie RNNs oder LSTMs weitgehend abgelöst. Sein zentrales Konzept ist der Self-Attention-Mechanismus, der es ermöglicht, Zusammenhänge zwischen beliebigen Positionen innerhalb einer Eingabesequenz effizient zu lernen.

Ein Transformer-Modell besteht typischerweise aus mehreren gestapelten Encoder- und/oder Decoder-Layern, wobei jeder Layer im Kern auf dem Multi-Head-Self-Attention-Mechanismus sowie nachgelagerten Feedforward-Schichten basiert. Die Self-Attention berechnet für jedes Token der Eingabe einen gewichteten Kontextvektor, indem alle anderen Token der Sequenz berücksichtigt werden. Dadurch werden sowohl lokale als auch globale Abhängigkeiten innerhalb der Sequenz effizient erfasst und verarbeitet. Die Attention-Funktion kann allgemein als eine Abbildung eines Query und einer Menge von Key-Value-Paaren auf einen Output beschrieben werden, wobei Query, Keys, Values und Output jeweils Vektoren sind. Der Output wird als gewichtete Summe der Values berechnet, wobei die Gewichtung durch eine Kompatibilitätsfunktion zwischen Query und Key bestimmt wird. Im Transformer wird hierzu die sogenannte *Scaled Dot-Product Attention* verwendet. Die Eingabe besteht aus Queries und Keys der Dimension d_k sowie Values der Dimension d_v . Für jeden Query wird das Skalarprodukt mit allen Keys berechnet, durch $\sqrt{d_k}$ skaliert und anschließend eine Softmax-Funktion angewendet, um die Gewichte zu bestimmen. Die Formel hierfür lautet:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2.1)$$

Hierbei werden die sogenannten Query-, Key- und Value-Matrizen aus der Eingabesequenz erzeugt und jeweils durch lineare Transformationen berechnet. Das Matrixprodukt QK^T ermöglicht es dem Modell, für jedes Token die Relevanz aller anderen Token explizit zu gewichten. Die Skalierung durch $\sqrt{d_k}$ verhindert, dass bei großen Wertebereichen der Softmax-Input zu hohe Varianzen aufweist, was zu

kleinen Gradienten und somit erschwerter Optimierung führen könnte. Um die Modellierungskapazität weiter zu erhöhen, wird im Transformer die Multi-Head Attention eingesetzt. Dabei werden mehrere Attention-Layer ("Heads") parallel ausgeführt, die jeweils eigene gewichtete Kombinationen der Eingabesequenz lernen. Die Ergebnisse der einzelnen Heads werden anschließend zusammengeführt und erneut linear transformiert. Dies ermöglicht dem Modell, verschiedene Aspekte der Beziehungen zwischen Tokens gleichzeitig zu erfassen und komplexere Muster zu erkennen.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \quad (2.2)$$

mit

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.3)$$

Die Projektionsmatrizen besitzen dabei folgende Dimensionen:

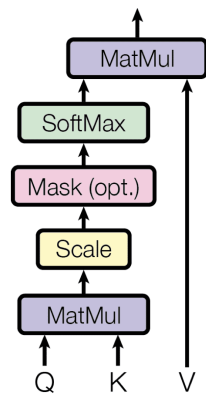
$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$$

$$W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$$

$$W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

Scaled Dot-Product Attention



Multi-Head Attention

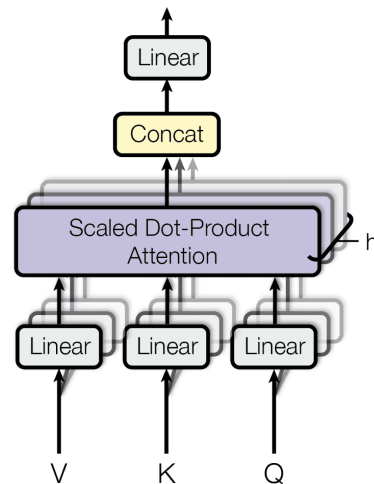


Abbildung 2.3.: Scaled Dot-Produkt Attention und Multi-Head Attention [18].

Die maximale Kontextlänge eines Transformers, also die Anzahl der Token, die das Modell in einem Durchgang berücksichtigen kann, ist eine zentrale architekturelle Eigenschaft. Sie wird maßgeblich durch die quadratische Skalierung der

Self-Attention-Operation limitiert, da sowohl der Rechen- als auch der Speicherbedarf mit $\mathcal{O}(n^2)$ in der Sequenzlänge n wachsen. Praktisch legen Modellentwickler die Kontextlänge im Vorfeld als festen Wert fest, der sich an den Hardwarekapazitäten und den Anforderungen der jeweiligen Anwendung orientiert. Eine Erhöhung der Kontextlänge über die Trainingskonfiguration hinaus ist meist nur eingeschränkt möglich [17].

Der Transformer hat sich insbesondere durch seine Flexibilität, Parallelisierbarkeit und Effektivität beim Lernen komplexer Abhängigkeiten als Standardarchitektur für NLP-Aufgaben etabliert. Moderne LLMs wie GPT, BERT oder T5 basieren alle auf Varianten des Transformers und nutzen dessen Fähigkeit, um Aufgaben aus den Bereichen NLU und NLG (vgl. Abschnitt 2.3) zu modellieren und effizient zu lösen [19].

2.4.3. Multimodales LLM GPT-4o

Im Mai 2024 wurde der Generative Pre-trained Transformer 4 Omni (GPT-4o) von OpenAI vorgestellt und repräsentiert einen bedeutenden Fortschritt im Bereich der multimodalen LLM. Im Gegensatz zu seinen Vorgängermodellen, die primär auf die Verarbeitung von Textdaten spezialisiert waren, ermöglicht GPT-4o die Integration und Verarbeitung unterschiedlicher Datenmodalitäten wie Text, Bild und Audio in beliebigen Kombinationen. Dies umfasst sowohl das Verarbeiten als auch das Generieren dieser Modalitäten [20].

Zudem weist GPT-4o eine deutliche Verbesserung hinsichtlich der Geschwindigkeit und Effizienz im Vergleich zu früheren Versionen auf. Darüber hinaus setzt das Modell neue Maßstäbe in Bezug auf Genauigkeit, Kontextverständnis und Flexibilität bei der Aufgabebearbeitung.

Die multimodale Architektur von GPT-4o erlaubt eine nahtlose Kontextverknüpfung zwischen den verschiedenen Datentypen. Durch die gleichzeitige Verwendung mehrerer Modalitäten als Ein- und Ausgabeparameter eröffnet GPT-4o neue Anwendungsszenarien, etwa im Bereich der Dokumentenverarbeitung, Barrierefreiheit sowie für komplexe Assistenzsysteme. Beispielsweise kann das Modell ein Bild analysieren, eine darin enthaltene Fragestellung in Textform erkennen, diese semantisch interpretieren und daraufhin eine passende Antwort in Textform geben [21]. Durch die Beschaffenheit multimodaler LLMs ist es möglich, Teil- oder Gesamtprozesse von klassischen Verfahren der Informationsextraktion wie OCR, OCR-Nachverarbeitung oder NER zu ersetzen [22]. Durch das zusätzliche Semantikverständnis, welches durch das LLM im Training erlernt wird, haben diese Systeme zusätzliche Fähigkeiten, welche sich von klassischen Verfahren deutlich abheben. So ist es möglich, das System mit benötigtem Kontext anzureichern, um so bessere Ergebnisse zu erzielen. Insgesamt stellt GPT-4o einen Paradigmenwechsel innerhalb der LLMs dar, da die Grenzen zwischen den einzelnen Datenmodalitäten zunehmend kleiner werden und eine multimodale, KI-gestützte Informationsverarbeitung ermöglicht wird [21].

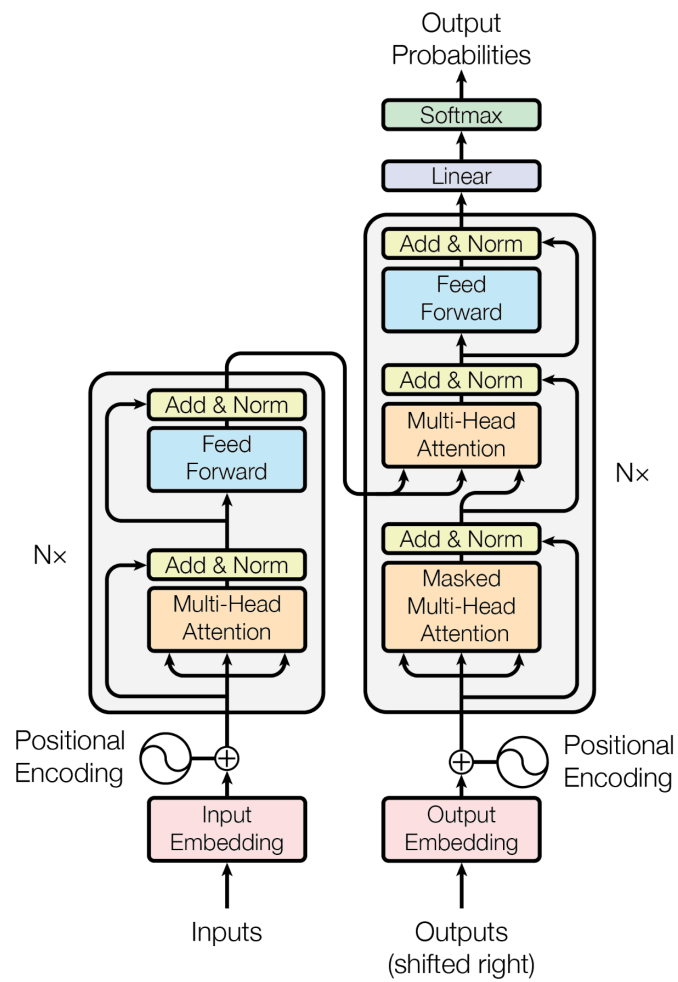


Abbildung 2.4.: Transformer-Modell mit Self-Attention-Mechanismus, bestehend aus Encoder, Decoder und Multi-Head Attention [18].

Außerdem stellt das Modell einen signifikanten Fortschritt im Bereich der natürlichen Sprachverarbeitung dar. Aufgrund seiner umfangreichen Trainingsdaten und fortgeschrittenen Architektur ist das Modell in der Lage, komplexe Textdaten zu analysieren, relevante Informationen präzise zu extrahieren und diese in strukturierter Form aufzubereiten. Dadurch eignet es sich insbesondere für Aufgaben der Informationsextraktion und -strukturierung in unstrukturierten Dokumenten. Im Kontext der Dokumentenverarbeitung ermöglicht dies beispielsweise die automatische Erkennung und Klassifikation von Entitäten, das Verständnis semantischer Zusammenhänge sowie die Generierung von Zusammenfassungen oder Antworten auf spezifische Fragestellungen. Dadurch wird die Grundlage für eine effiziente und weitgehend automatisierte Datenverarbeitung geschaffen [21].

2.4.4. Prompt Engineering

Als Prompt wird die Texteingabe eines LLMs bezeichnet, die in natürlicher Sprache erfolgt. Prompt Engineering ist ein zentraler Ansatz, um die Ausgabequalität von Sprachmodellen (LLMs) gezielt zu steuern und zu optimieren. Durch die systematische Gestaltung und Anpassung von Prompts kann nicht nur die Genauigkeit der Ergebnisse verbessert, sondern auch die Fehlerquote signifikant reduziert werden. Prompt Engineering ermöglicht es, die Fähigkeiten großer Sprachmodelle effektiv auszuschöpfen, indem Aufgabenstellungen so formuliert werden, dass das Modell relevante Informationen korrekt extrahiert, verarbeitet und während der Inferenz im Kontext verwendet. Die Entwicklung eines optimierten Prompts ist nicht trivial, da bei der Konstruktion verschiedene Faktoren zu berücksichtigen sind. Dazu zählen das vorherige Training des Modells sowie Aspekte wie Wortwahl, Stil, Ton, Struktur und Gesamtkontext. Ein schlecht konstruierter Prompt kann das Modell daran hindern, akkurate Ausgaben zu generieren. Der Aufbau eines Prompts wird in der Literatur zwischen *System Prompt*, *Role Prompt* und *Context Prompt* unterschieden.

Die Konfiguration der Ausgabe eines Sprachmodells (LLM) stellt einen zentralen Aspekt für das Prompt Engineering dar. Neben der Ausgestaltung des Prompts beeinflussen verschiedene Parameter die Art, Qualität und Konsistenz der generierten Texte. Zu den wichtigsten Konfigurationsparametern zählen beispielsweise die sogenannte Temperatur, die den Grad der Zufälligkeit bei der Textgenerierung steuert; eine niedrigere Temperatur sorgt dafür, dass das Ergebnis deterministischer wird. Zusätzlich gibt es die Parameter *Top-k*- und *Top-p*. Mit diesen zwei Parametern wird der Suchraum für mögliche nächste Tokens eingeschränkt. Darüber hinaus kann die maximale Länge der Ausgabe in Token eingestellt werden, was Rechenressourcen schont und Kosten spart.

Eine sorgfältige Konfiguration dieser Parameter ist entscheidend, um konsistente, relevante und den jeweiligen Anforderungen entsprechende Ergebnisse zu erzielen. Die richtige Balance zwischen Kreativität (Diversität der Ausgabe) und Zuverlässig-

keit (Reproduzierbarkeit und Genauigkeit) ist dabei abhängig von der jeweiligen Anwendungsdomäne und Aufgabe. In der Praxis wird die optimale Konfiguration oft empirisch durch Tests und iterativer Anpassung gefunden [2].

	Temperatur	Token Limit
Wert	0.1	5
Prompt	Beschreibe die Ware: Elektronische Bauteile.	
Ausgabe	Elektronische Bauteile sind kleine	

Tabelle 2.1.: Prompt mit geringer Temperatur von 0.1 und niedrigem Token-Limit von 5: Die Ausgabe ist sehr vorhersehbar und endet nach 5 Tokens.

	Temperatur	Top-P	Token Limit
Wert	0.9	0.5	20
Prompt	Beschreibe die Ware: Elektronische Bauteile.		
Ausgabe	Elektronische Bauteile sind wichtige Komponenten in zahlreichen technischen Geräten. Sie werden in Computern, Smartphones, aber auch in komplexen Maschinen verwendet und können sehr unterschiedlich sein.		

Tabelle 2.2.: Prompt mit hoher Temperatur von 0.9 und Top-P 0.5 mit einem hohen Token-Limit von 20: Die Ausgabe ist kreativer und ausführlicher.

Zero-shot Prompting

Unter Zero-shot Prompting versteht man die einfachste Art des Prompts. Hier wird der Prompt als einfache Beschreibung oder Aufgabenstellung beschrieben. Dies können einfache Fragen, Instruktionen oder zufälliger Text sein. Bei dieser Methode des Promptings werden dem Modell keine Beispiele zur Verfügung gestellt [2].

Prompt	Extrahiere die folgenden Entitäten aus dem untenstehenden Zolldokument: Versender, Empfänger, Warenbezeichnung, Zolltarifnummer, Menge. Dokument: „Der Versender: ABC Logistics GmbH, Empfänger: Müller Import Export AG, Warenbezeichnung: Elektronische Bauteile, Zolltarifnummer: 85423990, Menge: 500 Stück.“
Ausgabe	Versender: ABC Logistics GmbH; Empfänger: Müller Import Export AG; Warenbezeichnung: Elektronische Bauteile; Zolltarifnummer: 85423990; Menge: 500 Stück

Tabelle 2.3.: Beispiel eines Zero-Shot Prompts mit Parametereinstellung zur Extraktion von Entitäten aus Zolldokumenten

In-Context Learning (ICL)

Beim ICL wird der Prompt mit zusätzlichem Wissen angereichert. Dies wird entweder in Form von Beispielen (z. B. Aufgaben-Lösungs-Paare) oder durch die Integration von externem Wissen aus Dokumenten. Dadurch wird das LLM befähigt, während der Inferenz aus den im Prompt enthaltenen Informationen zu „lernen“, ohne die Modellgewichte, wie beim klassischen Training, zu verändern. Durch das Bereitstellen von zusätzlichem Kontext innerhalb des Prompts kann das Modell die Muster und Zusammenhänge erkennen und auf neue Aufgaben übertragen. Dieses Verhalten wird erst durch das vorherige Fine-Tuning eines LLM auf beispielbasierte Antwortgenerierung ermöglicht [23].

Insbesondere Few-Shot Learning, bei dem dem Modell ein oder mehrere Beispiele im Prompt zur Verfügung gestellt werden, erzielt in der Regel deutlich bessere Ergebnisse als Zero-Shot Prompting, bei dem lediglich eine Aufgabenstellung, aber keine Beispiele gegeben werden.

Prompt	<p>Beispiel 1: Text: „Der Versender: Exportfirma XY, Empfänger: Technik AG, Warenbezeichnung: Maschinen, Zolltarifnummer: 12345678, Menge: 10 Stück.“ Extrahierte Entitäten: Versender: Exportfirma XY; Empfänger: Technik AG; Warenbezeichnung: Maschinen; Zolltarifnummer: 12345678; Menge: 10 Stück</p> <p>Beispiel 2: Text: „Der Versender: Hans Spedition, Empfänger: Auto AG, Warenbezeichnung: Ersatzteile, Zolltarifnummer: 87654321, Menge: 20 Stück.“ Extrahierte Entitäten: Versender: Hans Spedition; Empfänger: Auto AG; Warenbezeichnung: Ersatzteile; Zolltarifnummer: 87654321; Menge: 20 Stück</p> <p>Neue Aufgabe: Text: „Der Versender: ABC Logistics GmbH, Empfänger: Müller Import Export AG, Warenbezeichnung: Elektronische Bauteile, Zolltarifnummer: 85423990, Menge: 500 Stück.“</p>
Ausgabe	Versender: ABC Logistics GmbH; Empfänger: Müller Import Export AG; Warenbezeichnung: Elektronische Bauteile; Zolltarifnummer: 85423990; Menge: 500 Stück

Tabelle 2.4.: In-Context Learning (ICL) mit Beispielen zur Extraktion von Entitäten aus Zolldokumenten

One-shot & Few-shot Learning

Diese Promptingmethoden sind Unterkategorien von ICL. Die Modellleistung wird bei diesen Verfahren durch zusätzliches Wissen im Kontextfenster optimiert. Das LLM wird sozusagen in eine gewisse Richtung gelenkt, und bekommt durch ein oder mehrere Aufgaben-Lösungs-Paare Ansätze, an eine Problemstellung heran zu gehen und diese in einem definierten Umfeld umzusetzen [24].

Prompt	<p>Extrahiere die Entitäten wie im Beispiel:</p> <p>Text: „Der Versender: Exportfirma XY, Empfänger: Technik AG, Warenbezeichnung: Maschinen, Zolltarifnummer: 12345678, Menge: 10 Stück.“</p> <p>Extrahierte Entitäten: Versender: Exportfirma XY; Empfänger: Technik AG; Warenbezeichnung: Maschinen; Zolltarifnummer: 12345678; Menge: 10 Stück</p> <p>Neue Aufgabe:</p> <p>Text: „Der Versender: ABC Logistics GmbH, Empfänger: Müller Import Export AG, Warenbezeichnung: Elektronische Bauteile, Zolltarifnummer: 85423990, Menge: 500 Stück.“</p>
Ausgabe	<p>Versender: ABC Logistics GmbH; Empfänger: Müller Import Export AG; Warenbezeichnung: Elektronische Bauteile; Zolltarifnummer: 85423990; Menge: 500 Stück</p>

Tabelle 2.5.: One-Shot/Few-Shot Prompting zur Extraktion von Entitäten aus Zolldokumenten

Instruktionsbasiertes Lernen

Beim instruktionsbasierten Lernen wird ein Large Language Model (LLM) durch eine explizite, natürlichsprachlich formulierte Anweisung dazu aufgefordert, eine bestimmte Aufgabe auszuführen. Im Gegensatz zu Few-Shot-Prompting werden dabei keine Beispiele vorgegeben, sondern die Zielaufgabe lediglich beschrieben. Diese Methode nutzt die Fähigkeit von modernen LLMs, Aufgaben allein auf Basis sprachlicher Instruktionen zu generalisieren, ohne dass eine Änderung der Modellparameter erforderlich ist.

Durch sogenanntes Instruction Tuning, das im Rahmen der Modellentwicklung auf großen Mengen an Aufgaben-Anweisungen erfolgt, wie z.B. bei FLAN oder Super-NaturalInstructions [19, 25], wird das Modell darauf vorbereitet, verschiedenartige Aufgabenstrukturen aus reinen Instruktionen heraus zu interpretieren. Dies ermöglicht eine flexible und anwenderfreundliche Nutzung, insbesondere bei standardisierten Aufgaben wie der Extraktion benannter Entitäten aus Texten.

Diese Methode ist besonders für domänenspezifische Extraktionsaufgaben geeignet, bei denen der Informationsbedarf klar definiert ist und sich mit einer einzelnen Anweisung erfassen lässt.

Prompt	Extrahiere die Entitäten Schritt für Schritt aus dem folgenden Zolldokument: Dokument: „Der Versender: ABC Logistics GmbH, Empfänger: Müller Import Export AG, Warenbezeichnung: Elektronische Bauteile, Zolltarifnummer: 85423990, Menge: 500 Stück.“ Schritt 1: Suche nach dem Versender. Schritt 2: Suche nach dem Empfänger. Schritt 3: Suche nach der Warenbezeichnung. Schritt 4: Suche nach der Zolltarifnummer. Schritt 5: Suche nach der Menge.
Ausgabe	Versender: ABC Logistics GmbH; Empfänger: Müller Import Export AG; Warenbezeichnung: Elektronische Bauteile; Zolltarifnummer: 85423990; Menge: 500 Stück

Tabelle 2.6.: instruktionsbasiertes Prompting mit Schrittweisen Anweisungen für Extraktion von Entitäten aus Zolldokumenten

Diese Methoden führen häufig zu einer signifikanten Verbesserung der generierten Ausgabesequenz. Darüber hinaus lassen sich die verschiedenen Prompting-Techniken – wie Instruction-Based Learning und In-Context Learning – gezielt miteinander kombinieren, um deren jeweilige Stärken zu vereinen. Hybride Promptingansätze haben sich in aktuellen Studien als besonders leistungssteigernd erwiesen und bilden somit eine wichtige Grundlage für den effektiven Einsatz von LLM in komplexen Extraktionsaufgaben [26, 27].

2.5. Definition der Zolldaten

Unter Zolldaten werden in dieser Arbeit spezifische, im internationalen Warenverkehr regelmäßig vorkommende Dokumentenarten verstanden. Zu den zentralen Zolldokumenten zählen insbesondere der Frachtbrief, das Zolldokument selbst, die Handelsrechnung, der Lieferschein, die Warenverkehrsbescheinigung sowie das Präferenzdokument. Diese Dokumente sind essenziell für die Abwicklung und Kontrolle grenzüberschreitender Warenbewegungen und enthalten für die Zollabfertigung relevante Informationen.

Einige dieser Dokumenttypen, wie beispielsweise der Frachtbrief oder das Zolldokument, folgen einem standardisierten Aufbau mit festgelegtem Schema, was ihre automatisierte Verarbeitung begünstigt. Andere Dokumente, etwa Rechnungen oder Lieferscheine, können hingegen je nach Aussteller in Format und Struktur erheblich variieren.

Im Folgenden werden die einzelnen Dokumentenarten näher beleuchtet und die Anforderungen, wie wichtige Entitäten, in Form eines Spaltenschemas definiert.

Hierbei werden die Schemas in zwei Kategorien unterteilt, das erste Schema stellt die Kopfdaten einer einzelnen Sendung dar, während das zweite Schema sich auf die Einzelpositionen pro Artikel innerhalb einer Sendung bezieht.

Dokumentenart	Felder	Layout
Handelsrechnung	Rechnungsnummer, Rechnungsdatum, Bestellnummer	Variabel
Lieferschein	Lieferscheinnummer, Lieferdatum, Empfänger, gelieferte Waren, Menge	Variabel
Frachtbrief	Packstückanzahl, Bruttogewicht, Versendungsdatum	Variabel
Zolldokument (T1)	Referenznummer (MRN), Zollsiegel (Seal[19 10]), Wareneingangsnr., Gültigkeit, Gesamtpositionen, Gesamtpakete, Gesamtgewicht	Strukturiert
Warenverkehrsbescheinigung (ATR)	Dokumenten-Nr., Aussteller, Ursprungsland, Bestimmungsland, Warenbezeichnung	Strukturiert
Präferenzdokument (EUR1)	Präferenznachweis-Nr., Aussteller, Ursprungsland, Begünstigte Waren, Referenz auf Handelsabkommen	Strukturiert

Tabelle 2.7.: Zentrale Zolldokumente mit Felder für die zu extrahierenden Entitäten der Kopfdaten einer Sendung und Grad des Layouts

Feld	Beschreibung
Rechnungsnummer	Eindeutige Kennung der ausgestellten Handelsrechnung
Rechnungsdatum	Ausstellungsdatum der Handelsrechnung
Bestellnummer	Interne oder externe Referenznummer der Bestellung
Lieferscheinnummer	Eindeutige Nummer zur Identifikation des Lieferscheins
Lieferdatum	Datum der Warenlieferung bzw. des Versands
Empfänger	Name und Adresse des Warenempfängers
Gelieferte Waren	Auflistung der im Lieferschein enthaltenen Warenpositionen
Menge	Gesamtmenge bzw. Summe der gelieferten Waren
Packstückanzahl	Anzahl der Packstücke, die transportiert werden
Bruttogewicht	Gesamtgewicht der Sendung inkl. Verpackung (Bruttomasse)
Versendungsdatum	Datum des Versands der Ware
Referenznummer (MRN)	Movement Reference Number, eindeutige Referenz für das Zolldokument T1
Zollsiegel (Seal[19 10])	Nummer und Kennzeichen des verwendeten Zollsiegels
Wareneingangsnummerstempel	Nummer zur Kennzeichnung des Wareneingangs beim Zoll
Gültigkeit	Zeitliche Gültigkeit oder Frist des Dokuments
Gesamtpositionen	Anzahl der im Dokument aufgeführten Einzelpositionen
Gesamtpakete	Gesamtzahl der in der Sendung enthaltenen Pakete
Gesamtgewicht	Brutto- oder Nettogewicht der gesamten Sendung
Dokumenten-Nr.	Offizielle Nummer des ausgestellten Dokuments (z. B. ATR)
Aussteller	Verantwortliche Organisation oder Person, die das Dokument ausstellt
Ursprungsland	Herkunftsland der Ware
Bestimmungsland	Zielland der Ware
Warenbezeichnung	Vollständige handelsübliche Bezeichnung der Ware
Präferenznachweis-Nr.	Nummer des Präferenzdokuments (z. B. EUR1)
Begünstigte Waren	Waren, für die eine Zollbegünstigung beantragt wird
Referenz auf Handelsabkommen	Bezug zu einem bestimmten Handelsabkommen, das eine Präferenz begründet

Tabelle 2.8.: Beschreibung der Felder der Kopfdaten

Feld	Beschreibung
Artikelnummer	Eindeutige Identifikationsnummer des Artikels
Preis	Preis der Einzelposition (numerisch)
Menge	Anzahl bzw. Menge der Warenposition (numerisch)
Menge_Maßeinheit	Maßeinheit der angegebenen Menge (z. B. kg, Stück)
Eigenmasse	Nettomasse des Artikels (numerisch)
Eigenmasse_Maßeinheit	Maßeinheit der Nettomasse (z. B. kg)
Anmelde- und handelsstatistische Menge	Statistische Zusatzmenge gemäß Zollanmeldung (numerisch)
Beantragte Begünstigung	Zollrechtlich beantragte Begünstigung (z. B. Präferenz)
Betriebliche Identifikationsnummer	Interne Identifikationsnummer des Unternehmens
Container-Nummer	Nummer des Containers bei Containertransport
Packstückzeichen und -nummer	Kennzeichnung und Nummerierung des Packstücks
Packstückanzahl	Anzahl der Packstücke (ganzzahlig)
Packstückart	Art des Packstücks (z. B. Karton, Palette)
Rohmasse_Maßeinheit	Maßeinheit der Bruttomasse
Ursprungsland	Ursprungsland der Ware (ISO-Code oder Klartext)
Warenbezeichnung	Beschreibung der Ware
Warennummer	Zolltarifnummer (HS-Code)
Referenz zu Dokument	Verweis auf ein zugehöriges Dokument (z. B. Rechnungsnummer)

Tabelle 2.9.: Felder der Einzelpositionen für die Extraktion aus Zolldokumenten

2.6. Evaluationsmetriken für die Bewertung des Systems

Die Bewertung der in dieser Arbeit erzielten Ergebnisse erfolgt durch den Vergleich eines hybriden Verfahrens, zweier klassischer Methoden sowie eines State-of-the-Art-Systems auf Basis eines LLM.

Ziel dieser Gegenüberstellung ist es, die Leistungsfähigkeit der Systeme hinsichtlich der Lösung eines Teilproblems des angestrebten Gesamtsystems objektiv zu beurteilen. Ein Vergleich des vollständigen Systems würde jedoch den Rahmen dieser Arbeit überschreiten.

Um die Genauigkeit und Qualität der verschiedenen Ansätze messbar und vergleichbar zu machen, werden klassische Metriken aus dem Bereich der Klassifikation angewandt. Dazu zählen Accuracy (Genauigkeit), Precision (Präzision), Recall (Sensitivität) sowie der F1-Score.

Accuracy (Genauigkeit): Bei der Accuracy auf Entitätsebene wird der Anteil der korrekt erkannten Entitäten (d. h. mit korrektem Typ und exakter Position) an allen im Goldstandard enthaltenen sowie vom System vorhergesagten Entitäten gemessen:

$$\text{Accuracy} = \frac{\text{Anzahl korrekter Treffer}}{\text{Gesamtanzahl Treffer}} \quad (2.4)$$

Precision (Präzision): Die Präzision beschreibt den Anteil der vom System extrahierten Entitäten, die tatsächlich korrekt sind:

$$\text{Precision} = \frac{\text{Anzahl korrekter Treffer}}{\text{Anzahl korrekter Treffer} + \text{Anzahl fälschlich erkanntter Entitäten}} \quad (2.5)$$

Hierbei steht die Anzahl fälschlich erkanntter Entitäten für die **False Positives**.

Recall (Sensitivität): Der Recall beschreibt den Anteil der tatsächlich relevanten Entitäten, die vom System erkannt wurden:

$$\text{Recall} = \frac{\text{Anzahl korrekter Treffer}}{\text{Anzahl korrekter Treffer} + \text{Anzahl übersehener Entitäten}} \quad (2.6)$$

Hierbei steht die Anzahl übersehener Entitäten für die **False Negatives**.

F1-Score: Der F1-Score ist das harmonische Mittel aus Präzision und Recall und stellt einen Kompromiss zwischen beiden Metriken dar:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.7)$$

Diese Metriken werden auf die extrahierten Entitäten angewendet und ermöglichen eine differenzierte Bewertung der Extraktionsergebnisse hinsichtlich Korrektheit und Vollständigkeit. Die Accuracy wird in dieser Arbeit bewusst als Recall-orientiertes Maß interpretiert, um sicherzustellen, dass alle relevanten Entitäten erkannt werden. Dies ist insbesondere im zollrechtlichen Kontext von zentraler Bedeutung, da unvollständige Extraktionen schwerwiegende Konsequenzen nach sich ziehen können [28].

Zur Bewertung des Gesamtsystems auf Klassenebene werden zusätzlich folgende aggregierte Metriken herangezogen, die eine gewichtete Betrachtung einzelner Klassen ermöglichen:

Support: Der Support bezeichnet die Anzahl der tatsächlichen Vorkommen einer Klasse im Datensatz. Er spielt eine implizite Rolle bei der Berechnung der Micro Metriken, da häufigere Klassen höher gewichtet werden:

$$\text{Support} = \text{korrekte Treffer}_c + \text{übersehene}_c \quad (2.8)$$

Micro-Accuracy: Die Micro-Accuracy aggregiert die korrekten Vorhersagen und die insgesamt betrachteten Fälle über alle Klassen hinweg:

$$\text{Micro-Accuracy} = \frac{\sum_c \text{korrekte Treffer}_c}{\sum_c (\text{korrekte Treffer}_c + \text{fälschlich erkannte}_c + \text{übersehene}_c)} \quad (2.9)$$

Micro-Precision: Die Micro-Precision aggregiert die korrekten und fälschlich erkannten Entitäten über alle Klassen hinweg:

$$\text{Micro-Precision} = \frac{\sum_c \text{korrekte Treffer}_c}{\sum_c (\text{korrekte Treffer}_c + \text{fälschlich erkannte}_c)} \quad (2.10)$$

Micro-Recall: Der Micro-Recall aggregiert die korrekten und übersehenen Entitäten über alle Klassen hinweg:

$$\text{Micro-Recall} = \frac{\sum_c \text{korrekte Treffer}_c}{\sum_c (\text{korrekte Treffer}_c + \text{übersehene}_c)} \quad (2.11)$$

Micro-F1: Der Micro-F1-Score basiert auf aggregierter Micro-Precision und Micro-Recall:

$$\text{Micro-F1} = 2 \cdot \frac{\text{Micro-Precision} \cdot \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}} \quad (2.12)$$

Macro-Precision: Die Macro-Precision ist der ungewichtete Durchschnitt der Präzi-

sion über alle Klassen:

$$\text{Macro-Precision} = \frac{1}{C} \sum_{c=1}^C \text{Precision}_c \quad (2.13)$$

Macro-Recall: Der Macro-Recall ist der ungewichtete Durchschnitt des Recalls über alle Klassen:

$$\text{Macro-Recall} = \frac{1}{C} \sum_{c=1}^C \text{Recall}_c \quad (2.14)$$

Macro-F1: Der Macro-F1-Score ist der ungewichtete Durchschnitt der F1-Scores über alle Klassen:

$$\text{Macro-F1} = \frac{1}{C} \sum_{c=1}^C \text{F1}_c \quad (2.15)$$

Auf die Macro-Accuracy wird in dieser Arbeit nicht näher eingegangen, da die Bewertung auf Klassenebene primär anhand der Micro-Metriken erfolgt. Für die abschließende Beurteilung der globalen Modellleistung wird ausschließlich der Macro-F1-Wert herangezogen. Dieser erlaubt eine ausgewogene Bewertung der Extraktionsqualität über alle Klassen hinweg, unabhängig von deren Verteilung im Datensatz. Zur Berechnung des Macro-F1 werden die Macro-Precision und der Macro-Recall benötigt, die jeweils als ungewichtetes Mittel der entsprechenden Werte über alle Klassen hinweg bestimmt werden.

Somit ist es möglich, Schwachstellen beim Erkennen einzelner Klassen zu identifizieren und gegebenenfalls zu optimieren [8].

Die in Zolldokumenten enthaltenen Entitäten können sowohl numerische Werte als auch Zeichenketten umfassen. Während bei numerischen Angaben die Bewertung der Extraktion auf Basis exakter Übereinstimmung erfolgen kann, gestaltet sich dies bei Zeichenketten schwieriger. Zur Bewertung der Ähnlichkeit extrahierter Zeichenfolgen werden daher häufig Metriken wie die Levenshtein-Distanz eingesetzt. Die Levenshtein-Distanz stellt einen grundlegenden Algorithmus im Bereich des Zeichenkettenvergleichs dar. Sie misst die Unähnlichkeit zwischen zwei Zeichenfolgen, indem sie die minimale Anzahl an einzelnen Zeichenoperationen berechnet, die erforderlich sind, um eine Zeichenkette in eine andere zu überführen. Zu diesen Operationen zählen das Einfügen, Löschen sowie das Ersetzen eines Zeichens.

Ein häufig zitiertes Beispiel ist die Umwandlung des Wortes „kitten“ in „sitting“, wofür genau drei Editieroperationen notwendig sind:

kitten → sitzen (Ersetzung von „k“ durch „s“)

sitten → sittin (Ersetzung von „e“ durch „i“)

sittin → sitting (Einfügen von „g“ am Ende)

Eine Editierung zählt als eines dieser Operationen – Einfügen, Löschen oder Ersetzen eines einzelnen Zeichens [29].

Um die erkannten Entitäten mit einer gewissen Toleranz zu bewerten, wird die Zeichenketten-Distanz mittels der Levenshtein-Distanz zwischen der erkannten Entität und dem korrespondierenden Label berechnet. Überschreitet diese Distanz einen definierten Schwellenwert nicht, so gilt die Entität als korrekt erkannt und wird als True Positive gewertet. Entitäten, die nicht erkannt oder falsch klassifiziert werden, fallen entsprechend in die Kategorien False Negative oder False Positive. Diese Einteilung bildet die Grundlage zur Berechnung der Metriken Accuracy, Precision, Recall und F1-Score.

Für die Berechnung der Levenshtein-Distanz wird in dieser Arbeit die Python-Bibliothek *thefuzz*⁴ verwendet, welche eine effiziente Implementierung des Algorithmus bereitstellt.

⁴<https://github.com/seatgeek/thefuzz>

3. Lösungsansatz zur Extraktion der Inhalte von Zolldokumenten

Klassische Extraktionspipelines, die auf der Kombination von OCR und NER basieren, sind in vielen industriellen Anwendungen etabliert und bieten eine solide Basis für strukturierte Informationsextraktion. Sie stoßen jedoch an Grenzen, sobald neue Entitätstypen auftreten und müssen daher oft explizit auf domänenspezifischen Anwendungsfällen trainiert oder nachtrainiert werden. Dies erfordert in der Regel sehr viele Trainingsdaten, die aufwändig vorbereitet und gelabelt werden müssen, um ein leistungsfähiges System zu erstellen. Moderne Ansätze auf Basis großer Sprachmodelle (LLM) ermöglichen eine flexible Textextraktion – vergleichbar mit OCR – sowie die Identifikation verschiedener Entitäten, ohne dass eine aufwändige Nachtrainierung oder explizite Merkmalsentwicklung erforderlich ist. Die Verarbeitung erfolgt dabei als durchgängige End-to-End-Lösung. Dies wird auch dadurch erreicht, dass die Modelle schon vorab auf sehr großen Datensätzen trainiert werden und die Modellfähigkeit zusätzlich durch weitere Optimierungsverfahren wie Reinforcement Learning from Human Feedback (RLHF) nachjustiert wird, wobei menschliches Feedback in die Gewichtsadjustierungen des LLMs mit einbezogen wird [30]. Durch diese Flexibilität können LLMs in sehr vielen Anwendungsbereichen eingesetzt werden, ohne viel Zeit in die Erstellung oder Vorbereitung von Testdaten zu investieren. Daher werden in dieser Arbeit beide Ansätze systematisch miteinander verglichen, um sowohl die Vorteile bewährter Methoden als auch das Potenzial aktueller LLM-Technologien für die Extraktion von Zolldaten zu evaluieren. Als Vergleichsbasis dient ein hybrider Ansatz, der Tesseract-OCR mit integrierter Layoutanalyse sowie spaCy-NER kombiniert. Beide Komponenten gelten als etablierte und robuste Werkzeuge im Bereich der NLP-basierten Informationsextraktion und haben sich in zahlreichen Studien als leistungsfähig erwiesen [7, 13].

Wie bereits in Abschnitt 1.1 erläutert, kann die Generalisierbarkeit von Sprachmodellen (LLMs) selbst bei großen Trainingsdatenmengen an ihre Grenzen stoßen – insbesondere bei sehr spezifischen Anwendungsfällen und in hochspezialisierten Domänen. Um dieses Problem zu umgehen, ist es möglich, den in Unterabschnitt 2.4.2 erwähnten Modellkontext (Aufmerksamkeitsschicht) der Transformerarchitektur zu nutzen, um die Modellleistung zu steigern [17]. Seit der Entwicklung der LLM sind zahlreiche Methoden entstanden, um die Leistungsfähigkeit dieser Modelle nicht durch explizites Nachtraining der Modellgewichte, sondern durch Kontextanreicherung während der Inferenz zu verbessern. Eine zentrale Methodik ist das Promptengineering (vgl. Unterabschnitt 2.4.4), bei dem das Modell mithilfe gezielt

gestalteter Eingabeaufforderungen (Prompts) dazu angeregt wird, Aufgaben in bestimmter Weise zu lösen oder bestimmte Informationen zu extrahieren. Besonders hervorzuheben sind das In-Context Learning, bei dem das Modell durch das Bereitstellen weniger Beispiele im Prompt („Few-Shot Learning“) in die Lage versetzt wird, neue Aufgaben ohne zusätzliche Trainingsdaten zu bearbeiten, oder auch das instruktionsbasierte Lernen, bei dem das Modell die Aufgabenstellung in einer bestimmten Abfolge von Schritten lösen soll. Die genannten Methoden haben sich als äußerst wirkungsvoll erwiesen, da sie die Anpassungsfähigkeit und Generalisierungsfähigkeit von LLMs erheblich erhöhen und somit den Einsatz in heterogenen und dynamischen Anwendungsszenarien, wie der Extraktion von Zolldaten, ermöglichen [31, 32].

Um das hybride System, bestehend aus OCR und NER, durch ein LLM-basiertes System abzubilden, wird die Aufgabe in zwei Teilaufgaben unterteilt. Im ersten Schritt werden die PDF-Dateien pro Sendung eingelesen und anschließend in Bildformate umgewandelt, da es sich häufig um gescannte oder fotografierte Dateien handelt. Dies ist notwendig, da MLLM direkt auf Bilddaten arbeiten und so in der Lage sind, durch gezieltes Prompting Entitäten direkt aus dem visuellen Dokumentkontext heraus zu extrahieren und von Bild zu Text vorherzusagen (siehe Implementierung (vgl. Listing A.1)). Dieser Teilschritt vereint die Aufgaben des OCR- und NER-Systems in einem einzigen Verarbeitungsschritt und ermöglicht dadurch sowohl die Extraktion der relevanten Entitäten als auch die Zuordnung der korrekten Attribut-Wert-Paare. Im klassischen, hybriden Ansatz müsste letzteres als zusätzlicher, oft sehr aufwändiger Verarbeitungsschritt separat implementiert werden. Ausgabe dieses Schrittes sind die Entitäten in Form einer JSON-Datei.

Im zweiten Schritt werden die nun teilstrukturierten Daten mithilfe des LLM in ein vollständig strukturiertes Format überführt. Dazu erhält das Sprachmodell sowohl die im vorherigen Schritt extrahierten Attribut-Wert-Paare im JSON-Format als Kontext, als auch einen spezifischen Aufgabenprompt. Die Ausgabe des Modells erfolgt wiederum im JSON-Format, wobei eine Schema-Prüfung durch vordefinierte Vorlagen erfolgt. Diese Vorlagen dienen dazu, dass das Modell die Ausgabe entsprechend des gewünschten Zielschemas generiert und im richtigen Format parst. Die Abbildung 3.1 zeigt die High-Level-Darstellung der entwickelten Datenpipeline zur Extraktion von Entitäten aus Zolldokumenten. Der schematische Ablauf illustriert die zentralen Verarbeitungsschritte von der Einlesung der PDF-Dateien, über die Konvertierung in Bilddaten und die anschließende Entitätsextraktion mittels multimodaler LLM, bis hin zur strukturierten Ausgabe der extrahierten Informationen.

Die Leistungsfähigkeit des eingesetzten Modells wurde im Rahmen dieser Arbeit gezielt durch den Einsatz unterschiedlicher Prompting-Techniken optimiert. Zur Sicherstellung einer hohen Reproduzierbarkeit wurde der Temperatur-Parameter, der das Maß an Zufälligkeit bei der Textgenerierung beeinflusst, standardmäßig auf den deterministischen Wert 0 gesetzt (vgl. Unterabschnitt 2.4.4). Die Effektivität multimodaler Large Language Models (LLMs) hängt maßgeblich von der

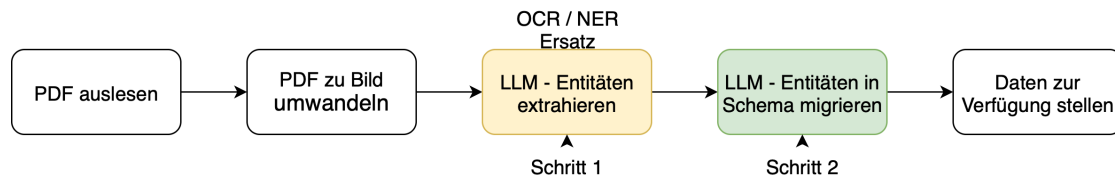


Abbildung 3.1.: Schematische Übersicht der Datenpipeline zur Entitätsextraktion aus Zolldokumenten. Die Abbildung illustriert die zentralen Verarbeitungsschritte durch ein multimodales LLM

Qualität der verwendeten Prompts ab. In dieser Arbeit wurden sechs Prompting-Techniken implementiert (vgl. Kapitel 4, Abschnitt A.3, Abschnitt A.2). Die Evaluierung der entwickelten Prompttechniken erfolgte anhand definierter Metriken (vgl. Abschnitt 2.6), wobei insbesondere die Extraktionsgenauigkeit sowie die Robustheit gegenüber variierenden Dokumentstrukturen analysiert wurden. Da für den Vergleich der unterschiedlichen Ansätze zur Extraktion von Entitäten aus Zolldokumenten eine objektive Bewertung der Modellleistung erforderlich ist, wird ein gelabelter Datensatz benötigt. Die manuelle Annotation dient dabei als Grundwahrheit, anhand dessen die Extraktionsgenauigkeit sowohl der klassischen als auch der LLM-basierten Methoden gemessen werden kann. Im Rahmen dieser Arbeit wurden die relevanten Entitäten in einer Stichprobe von Zolldokumenten nach einem vorab definierten einheitlichen Schema (vgl. Abschnitt 2.5) markiert. Die Annotation erfolgte manuell, wobei die Entitätenklassen berücksichtigt wurden. Um die Qualität der Annotation sicherzustellen, wurden die Daten zunächst von einem Annotator gelabelt und anschließend mit dem Fachbereich für Zollprozesse abgeglichen. Im Falle von Unstimmigkeiten erfolgte eine Klärung im Konsens. Der so entstandene Datensatz bildet die Grundlage für die Evaluierung und den Vergleich der Modellansätze.

4. Implementierung

Das System ist in *Python 3.9*⁵ auf der Plattform *Dataiku*⁶ implementiert (schematisch dargestellt in Anhang C) und nutzt die Azure OpenAI API, um sicherzustellen, dass keine übermittelten Daten in das Trainingsmodell zurückfließen. Die Datenverarbeitung erfolgt lokal, während die Modellanfragen über die Cloud-basierte API abgewickelt werden. Im Folgenden werden die zentralen Komponenten der Implementierung dargestellt, insbesondere die Integration der OCR- und NER-Komponenten, die Verarbeitungspipeline zur Extraktion der Zolldaten sowie das verwendete Prompt-Design. Die vollständige Implementierung ist im Abschnitt A.1 dokumentiert.

Die erste Teilaufgabe umfasst die Komponenten OCR und NER innerhalb der Extraktionspipeline. Zu diesem Zweck wird dem Modell ein Bild übergeben, das zuvor mit der Python-Bibliothek *Pillow*⁷ aus dem ursprünglichen PDF-Dokument extrahiert wurde. Durch diese Vorgehensweise kann die multimodale Verarbeitungsfähigkeit des LLM genutzt werden, sodass das Modell in der Lage ist, die visuellen Informationen zu interpretieren und die enthaltenen Entitäten in Textform zu extrahieren. Die Rückgabe erfolgt im JSON-Format und beinhaltet sämtliche im Dokument erkannten Entitäten.

```
1 # LLM API Call Part 1
2 response = client.chat.completions.create(
3     model=AZURE_OPENAI_MODEL,
4     response_format={ "type": "json_object" },
5     messages=[
6         {
7             "role": "system",
8             "content": system_prompt.format(url=url)
9         },
10        {
11            "role": "user",
12            "content": [
13                {"type": "text", "text": user_prompt},
14                {
15                    "type": "image_url",
16                    "image_url": {
```

⁵<https://www.python.org/downloads/release/python-390/>

⁶<https://www.dataiku.com>

⁷<https://pillow.readthedocs.io/en/stable/index.html>

```

17         "url": f"data:image/png;base64,{base64_image}",
18         "detail": "high"
19     }
20 }
21 ]
22 }
23 ],
24 temperature=0.0,
25 )
26 return response.choices[0].message.content

```

Listing 4.1: LLM als OCR-Ersatz – PDF zu Bild und Bild zu Text Extraktion

Nachdem die in Teilaufgabe eins extrahierten Entitäten persistiert wurden, erhält das LLM im Rahmen von Teilaufgabe zwei diese Entitäten als kontextuelle Information. Ergänzt wird dieser Kontext durch ein strukturelles Zielschema im JSON-Format sowie eine textuelle Prompt-Anweisung. Auf Basis dieser Eingaben generiert das Modell die Zielausgabe in Form einer JSON-Datei (vgl. Listing 4.2). Die Einhaltung des vorgesehenen Ausgabeformats wird durch einen nachgelagerten Parser gewährleistet, welcher die Modellantwort anhand eines vordefinierten Schemas validiert. Diese Validierungsfunktionalität wird über die Azure API bereitgestellt. Der Parser nutzt hierzu ein referenziertes Schemainstanzmodell, welches auch verschachtelte Datenstrukturen überprüfen kann (vgl. Listing 4.3).

```

1  # LLM Aufruf Teilaufgabe zwei
2  response = client.beta.chat.completions.parse(
3      model=AZURE_OPENAI_MODEL,
4      messages=[
5          {
6              "role": "system",
7              "content": system_prompt
8          },
9          {
10             "role": "user",
11             "content": [
12                 {
13                     "type": "text",
14                     "text": user_prompt.format(
15                         json_raw=json_raw,
16                         schema_json=final_schema
17                     )
18                 }
19             ]
20         }
21     ],
22     response_format=ZollSchema,
23     temperature=0.0,
24 )
25 return response.choices[0].message.parsed

```

Listing 4.2: Schemamigration des extrahierten JSON-Contents aus Teilaufgabe eins

Um Anforderungen der Ausgabe zu gewährleisten, werden Modelle in Form von Schlüssel-Wert-Tupeln definiert. Diese können beliebig verschachtelt werden und Werte der Tuple können wiederum ein Modell beinhalten. Ein Modell kann wie folgt aussehen.

```
1 class ZollSchema(BaseModel):
2     document_data: Zollkopfdaten
3     goods_positions: List[Zolleinzelpositionen]
4     sendung: str
5     token_count_doc: int
6     prompt_version: str
```

Listing 4.3: Pydantic Modell für eine Schemaüberprüfung

Ein zentraler Bestandteil der Implementierung stellt das gezielte Design der Prompts zur Steuerung des Verhaltens des eingesetzten LLMs dar.

Im Rahmen der Entwicklung wurden sechs unterschiedliche Prompting-Strategien evaluiert, die sich hinsichtlich Struktur, Grad der Beispielhaftigkeit und Kontexttiefe unterscheiden. Dazu zählen klassische Ansätze wie Zero-Shot- und One-Shot-Prompting sowie kombinierte Verfahren, die instruktionsbasiertes Lernen (Instruction-Based Learning (IBL)) mit Beispielen verknüpfen.

Aufgrund des Umfangs der jeweiligen Promptformulierungen wird in diesem Kapitel ausschließlich die Zero-Shot-Methode für beide Teilaufgaben exemplarisch dargestellt. Die übrigen Varianten – Kombinationen aus One-Shot und IBL sowie Few-Shot und IBL – bauen auf den hier beschriebenen Grundstrukturen auf und ergänzen diese durch zusätzliche Beispiele. Eine vollständige Übersicht über sämtliche Promptformulierungen findet sich im Anhang (Abschnitt A.2, Abschnitt A.3). Die im Folgenden dargestellten Prompts verwenden das Zero-Shot-Verfahren, bei dem das LLM lediglich eine natürliche Spracheingabe in Form einer Anweisung erhält, jedoch ohne explizite Beispiele oder Zwischenschritte. Die Formulierung der Prompts erfolgte iterativ und wurde anhand empirischer Testergebnisse hinsichtlich Genauigkeit, Robustheit gegenüber OCR-bedingten Fehlern sowie Antwortkonsistenz optimiert.

Jeder Prompt besteht aus zwei funktionalen Komponenten: einem System-Prompt und einem User-Prompt. Der System-Prompt definiert den globalen Rahmen sowie die Rollenzuweisung für das Modell – etwa die Aufgabe als „Experte für Dokumentenanalyse“ – und legt das erwartete Antwortformat fest. Der User-Prompt enthält die eigentliche Instruktion und ggf. den Eingabetext (z. B. extrahierte Entitäten oder das OCR-Ergebnis). Zusammen steuern beide Komponenten die Verarbeitung und Ausgabe des Modells.

Im Folgenden sind zwei Prompt-Szenarien für unterschiedliche Aufgabenbereiche des Systems aufgelistet:

- **Prompt 1: Datenextraktion aus Zolldokumenten**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and transport documents. There are six main types of documents: - Waybill (Frachtbrief) - T1 Transit Document (Zolldokument) - Invoice (Rechnung) - Delivery Note (Lieferschein) - Movement Certificate (ATR) - Movement Certificate (EUR1)

Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making up data if not available write `None` as value. Always provide the key `"referenz zu dokument": {url}`

User Prompt:

Extract all relevant fields in the customs document in JSON format. Use original language for the values. If the document has no relevant data, return an empty JSON object.

- **Prompt 2: Transformation von JSON-Daten in Ziel-Schema**

System Prompt:

You are a data transformation tool that takes in JSON data and a reference JSON schema, and outputs JSON data according to the schema. Not all of the data in the input JSON will fit the schema, so you may need to omit some data or add null values to the output JSON. Translate all data into German if not already in German. Ensure values are formatted as specified in the schema (e.g. dates as YYYY-MM-DD). Here is the schema: {schema json}.

User Prompt:

Transform the following raw JSON data according to the provided schema. Ensure all data is in English and formatted as specified by values in the schema. Here is the raw JSON: {json raw}.

5. Evaluierung des Systems

Für die Evaluierung und Bewertung des Systems wurden pro Prompttechnik definierte Testszenarien durchgeführt, anhand derer die Extraktionsleistung systematisch analysiert wurde. Für jede Prompttechnik wurde die Version getrackt, um Genauigkeit des Systems über alle verwendeten Methoden zu dokumentieren. Bei der Systemgegenüberstellung des klassischen Systems zu dem LLM-basierten System wurden die Entitäten keiner Klasse zugeordnet. Da im Rahmen dieser Arbeit kein Modell neu trainiert wurde, sondern auf bereits vortrainierte Modelle zurückgegriffen wurde, beschränkte sich die Evaluation auf eine binäre Klassifikation. Es wurde zwischen den Fällen „Entität gefunden“ und „Entität nicht gefunden“ unterschieden. Eine feinere Klassifikation nach Entitätstypen hätte ein Finetuning des Modells oder die Definition regelbasierter Extraktionslogik erfordert, wofür jedoch ein großer annotierter Datensatz notwendig gewesen wäre.

Die Bewertung der Extraktionsleistung basierte auf zwei unterschiedlichen Vergleichsverfahren, abhängig vom Typ der extrahierten Entität. Numerische Werte (z.B. Mengen, Preise oder Gewichtseinheiten) wurden auf Basis exakter Übereinstimmung mit den Referenzwerten geprüft. Eine numerische Extraktion wurde dabei nur dann als korrekt gewertet, wenn der erkannte Wert exakt dem erwarteten Zielwert entsprach. Für Zeichenketten (z.B. Produktbezeichnungen, Zolllarifnummern oder Ursprungsangaben) kam ein Vergleich mittels Levenshtein-Distanz (vgl. Abschnitt 2.6) zur Anwendung. Dieses Verfahren erlaubt eine gewisse Toleranz gegenüber kleineren OCR-Fehlern oder typografischen Abweichungen, ohne die strukturelle Korrektheit der Extraktion wesentlich zu beeinträchtigen [33]. Dabei wurde ein Schwellenwert (Threshold) von 90% gewählt, sodass erkannte Entitäten als korrekt klassifiziert wurden, wenn ihre Ähnlichkeit zum Zielwert mindestens 90% betrug. Die Auswahl dieses Schwellenwerts erfolgte, da die Metriken Micro-Accuracy, Micro-Recall und Micro-F1 im Bereich von 85% bis 100% kaum Schwankungen zeigten, während die Micro-Precision insbesondere bei den Kopfdaten ab einem Schwellenwert von mehr als 90% signifikant abnimmt (vgl. Abbildung 5.1, Abbildung 5.2). Diese Festlegung basiert auf empirischen Analysen der entsprechenden Metriken.

Durch die im ersten Teilschritt erfolgte Extraktion sämtlicher potenzieller Entitäten mittels des LLMs ergibt sich ein relativ niedriger Accuracy-Wert von 0,17 und ein Precision-Wert von maximal 0,25. Im Vergleich dazu weist der zweite Teilschritt eine etwas höhere Accuracy von 0,25 und eine signifikant höhere Präzision mit

einem Maximalwert von 0,56 auf (vgl. Tabelle 5.1, Tabelle 5.2). Die Ursache für die geringere Präzision im ersten Schritt liegt in der begrenzten Kontextinformation, die dem Modell hinsichtlich des gewünschten Ausgabeformats zur Verfügung stand. Dies führte zu einer Vielzahl von falsch-positiven Entitäten. Diese Entitäten waren zwar semantisch korrekt und im Dokument vorhanden, jedoch im Kontext des spezifischen Dokumententyps bzw. des zugehörigen Zollprozesses nicht relevant.

In Bezug auf den Recall zeigt sich, dass im ersten Teilschritt mit einem Höchstwert von 0,55 eine deutlich höhere Erkennungsrate erzielt wurde als im zweiten Schritt, in dem maximal 0,32 erreicht wurde. Allerdings liegt auch dieser Recall-Wert außerhalb eines für reale Zollanwendungen akzeptablen Bereichs, da die Vielzahl irrelevanter Entitäten zu einer geringeren praktischen Verwertbarkeit der Ergebnisse führt.

Der resultierende F1-Score beträgt 0,3 im ersten und 0,4 im zweiten Teilschritt. Dies verdeutlicht, dass die Strategien beider kombinierter Ansätze derzeit noch nicht die erforderliche Extraktionsgenauigkeit für eine produktive Nutzung im Zollkontext erreichen. Die Ergebnisse unterstreichen die Notwendigkeit, den semantischen Kontext sowie die Formatierungsvorgaben stärker im Prompt-Design zu berücksichtigen, um die Präzision ohne signifikanten Verlust an Recall zu erhöhen.

TP	FP	FN	Accuracy	Precision	Recall	F1	Methode
877	3418	725	0.17	0.2	0.55	0.3	One-Shot
870	2679	732	0.2	0.25	0.54	0.34	Few-Shot
861	2948	741	0.19	0.23	0.54	0.32	One-Shot + IBL
852	2854	750	0.19	0.23	0.53	0.32	Few-Shot + IBL
624	3605	978	0.12	0.15	0.39	0.21	Zero-Shot
616	3898	986	0.11	0.14	0.38	0.2	IBL

Tabelle 5.1.: Auswertung der aggregierten Metriken - absteigend sortiert nach Recall
- für alle Dokumente Promptmethode für die Extraktion durch Gpt-4o
in Teilschritt eins

TP	FP	FN	Accuracy	Precision	Recall	F1	Methode
518	469	1084	0.25	0.52	0.32	0.4	IBL
401	310	1201	0.21	0.56	0.25	0.35	Zero-Shot
172	307	1430	0.09	0.36	0.11	0.17	One-Shot + IBL
162	376	1440	0.08	0.3	0.1	0.15	One-Shot
155	317	1447	0.08	0.33	0.1	0.15	Few-Shot + IBL
142	275	1460	0.08	0.34	0.09	0.14	Few-Shot

Tabelle 5.2.: Auswertung der aggregierten Metriken - absteigend sortiert nach Recall - für alle Dokumente pro Promptmethode für die Schemamigration der in Schritt eins gefundenen Entitäten durch Gpt-4o in Teilschritt zwei

Im Vergleich dazu weist der Ansatz, der auf einer Kombination aus der Extraktion der Dokumenteninhalte durch Tesseract-OCR und der Entitätserkennung durch spaCy-NER basiert, eine deutlich geringere Extraktionsqualität auf (vgl. Tabelle 5.3). Die Analyse der Ergebnisse zeigt, dass die OCR-Komponente eine vergleichsweise hohe Fehlerquote aufweist, was insbesondere bei komplexen Layouts und geringer Scanqualität zu fehlerhaften oder unvollständigen Texteingaben führt. Infolgedessen ist auch die Qualität der nachgelagerten NER-Erkennung beeinträchtigt. Zwar identifiziert spaCy im Vergleich zum LLM eine größere Anzahl an Entitäten, was an den falsch positiven Werten zu erkennen ist, jedoch ist der Anteil korrekt erkannter und relevanter Entitäten äußerst gering. Dies spiegelt sich deutlich in den Evaluationsmetriken wider: Die Accuracy liegt bei 0,06, die Precision bei lediglich 0,07, während ein Recall von 0,28 erreicht wird. Der daraus resultierende F1-Score beträgt 0,11. Diese Werte verdeutlichen, dass der klassische OCR-NER-Ansatz ohne Feinjustierung des Modells auf die spezifischen Zolldaten in der vorliegenden Anwendung weder hinreichend präzise, noch robust genug ist, um den Anforderungen an eine zuverlässige und kontextbezogene Extraktion von Zolldaten gerecht zu werden. Der Ansatz wurde ausschließlich im Rahmen von Teilschritt eins evaluiert, da eine Vorhersage der Entitäten gemäß dem definierten Zollschemata mit regelbasierten NER-Modellen wie spaCy – ohne explizite Modellierung zollspezifischer Entitätstypen – nicht möglich ist, eine für Teilschritt zwei benötigte Vergleichsbasis abzubilden.

TP	FP	FN	Accuracy	Precision	Recall	F1
447	5853	1155	0.06	0.07	0.28	0.11

Tabelle 5.3.: Auswertung spaCy NER für die Extraktion Teilschritt eins

Die Bewertung des LLM-basierten Systems erfolgte durch den Abgleich der finalen Ausgabe mit einem speziell für den Zollanwendungsfall definierten Ausgabeschema. Dabei wurden sowohl die Kopfdaten als auch die Einzelpositionen pro Sendung analysiert (vgl. Tabelle 2.8). Neben den klassischen Metriken Precision, Recall und F1-Score auf Klassenebene wurden zusätzlich gewichtete Micro- und Macro-Durchschnitte berechnet. Diese ergänzenden Kennzahlen ermöglichten eine differenzierte Analyse der Extraktionsleistung einzelner Entitätsklassen und erlauben Rückschlüsse auf spezifische Schwächen. Dadurch lassen sich gezielte Nachjustierungen in zukünftigen Systemen ableiten und die Robustheit gegenüber domänenspezifischen Herausforderungen verbessern.

Bei den Kopfdaten aller 30 Sendungen wurde im Zero-Shot-Modus ein Bestwert von 0,33 für die Micro-Precision, 0,14 für den Micro-Recall und 0,20 für den Micro-F1-Score erzielt. Das bedeutet, dass lediglich 14% der benötigten Entitäten korrekt einer passenden Klasse zugeordnet wurden.

Prompttyp	TP	FP	FN	Mic.-Acc.	Mic.-Prec.	Mic.-Rec.	Micro-F1
IBL	54	128	308	0.11	0.3	0.15	0.2
Zero-Shot	51	104	311	0.11	0.33	0.14	0.2
Few-Shot + IBL	28	208	334	0.05	0.12	0.08	0.09
One-Shot	23	247	339	0.04	0.09	0.06	0.07
Few-Shot	22	202	340	0.04	0.1	0.06	0.08
One-Shot + IBL	23	223	339	0.04	0.09	0.06	0.08

Tabelle 5.4.: Micro-basierte Metriken zur Bewertung der Kopfdatenextraktion

Prompttyp	Macro-Precision	Macro-Recall	Macro-F1
IBL	0.18	0.09	0.17
Zero-Shot	0.19	0.09	0.19
Few-Shot + IBL	0.09	0.12	0.09
One-Shot	0.04	0.12	0.08
Few-Shot	0.06	0.07	0.07
One-Shot + IBL	0.05	0.07	0.07

Tabelle 5.5.: Macro-basierte Metriken zur Bewertung der Kopfdatenextraktion

Für die Einzelpositionen aller 30 Sendungen pro Promptversion wurden im Zero-Shot-Modus eine maximale Micro-Precision von 0,69, ein Micro-Recall von 0,53 sowie ein Micro-F1-Score von 0,60 erzielt. Das bedeutet, dass 60% der benötigten Entitäten über alle Klassen hinweg korrekt extrahiert wurden. Im Vergleich zu den

Ergebnissen der Kopfdaten stellt dies eine signifikante Verbesserung dar. Dennoch reicht diese Leistung nicht aus, um die notwendige Fehlertoleranzschwelle für kritische Zollprozesse zu erfüllen.

Prompttyp	TP	FP	FN	Mic.-Acc.	Mic.-Prec.	Mic.-Rec.	Micro-F1
Zero-Shot	214	97	188	0.43	0.69	0.53	0.60
IBL	205	171	285	0.31	0.55	0.42	0.47
One-Shot + IBL	17	25	40	0.21	0.4	0.3	0.34
One-Shot	30	66	92	0.16	0.31	0.25	0.28
Few-Shot + IBL	9	26	39	0.12	0.26	0.19	0.22
Few-Shot	6	21	30	0.11	0.22	0.17	0.19

Tabelle 5.6.: Micro-basierte Metriken zur Bewertung der Extraktion der Einzelpositionen

Prompttyp	Macro-Precision	Macro-Recall	Macro-F1
Zero-Shot	0.41	0.34	0.35
IBL	0.4	0.29	0.31
One-Shot + IBL	0.18	0.22	0.19
One-Shot	0.14	0.17	0.15
Few-Shot + IBL	0.12	0.12	0.12
Few-Shot	0.11	0.11	0.11

Tabelle 5.7.: Macro-basierte Metriken zur Bewertung der Extraktion der Einzelpositionen

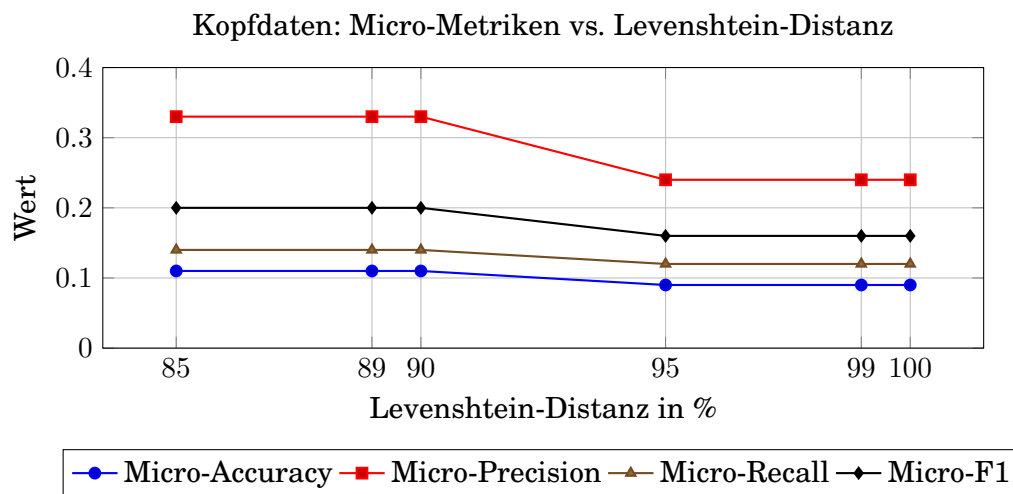


Abbildung 5.1.: Kopfdaten: Micro-Metriken in Abhängigkeit der Levenshtein-Distanz

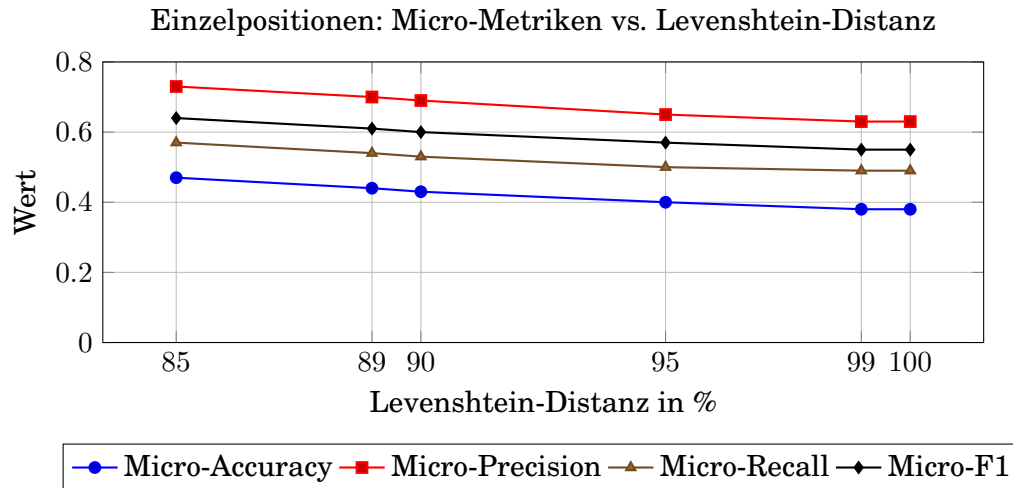


Abbildung 5.2.: Einzelpositionen: Micro-Metriken in Abhängigkeit der Levenshtein-Distanz

6. Zusammenfassung und Ausblick

In diesem Kapitel werden die zentralen Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf zukünftige Forschungs- und Entwicklungsmöglichkeiten gegeben. Nach einer kompakten Darstellung der wesentlichen Erkenntnisse aus den empirischen Analysen und der Evaluation des entwickelten Systems werden darauf aufbauend Potenziale und Herausforderungen für weiterführende Arbeiten sowie mögliche Anwendungsperspektiven aufgezeigt. Ziel ist es, die erreichten Fortschritte im Kontext der automatisierten Informationsextraktion aus Zolldokumenten kritisch zu reflektieren und Ansatzpunkte für eine kontinuierliche Weiterentwicklung aufzuzeigen.

6.1. Erreichte Ergebnisse

Im Rahmen der Evaluation zeigte sich, dass die Extraktionsgenauigkeit auf Ebene der Einzelpositionen deutlich höher ausfiel als bei den Kopfdaten (vgl. Tabelle 5.6, Tabelle 5.4). Insbesondere die Wahl des Schwellenwerts für die Levenshtein-Distanz hatte einen maßgeblichen Einfluss auf die erzielten Metriken: Während die Metriken Micro-Accuracy und Micro-Recall sowie der Micro-F1 über einen weiten Schwellenwertbereich weitgehend stabil blieben, war bei den Kopfdaten ab einem Schwellenwert von mehr als 90% ein deutlicher Rückgang der Precision zu beobachten (vgl. Abbildung 5.2, Abbildung 5.1). Die besten Ergebnisse hinsichtlich Micro-Accuracy und Micro-F1 wurden für die Einzelpositionen bei Schwellenwerten größer 90% erzielt.

Die Ergebnisse verdeutlichen zudem, dass die Leistungsfähigkeit des Systems maßgeblich von der Qualität und Struktur der zugrunde liegenden Daten beeinflusst wird. Insgesamt zeigen die empirischen Analysen, dass die Extraktionspipeline derzeit noch nicht ausreichend robust ist, um zollkritische Prozesse vollumfänglich und automatisiert abzulösen.

6.2. Ausblick

Durch die aktuell unzureichende Extraktionsgenauigkeit, insbesondere bei den Kopfdaten, sowie die beobachteten Schwankungen in den erzielten Metriken be-

steht weiterhin Optimierungsbedarf im Gesamtsystem. Zukünftige Arbeiten sollten sich daher auf die Verbesserung der Extraktionsleistung und der Robustheit der Extraktionspipeline konzentrieren. Insbesondere könnten weiterführende Untersuchungen die Verwendung unterschiedlicher Ähnlichkeits- und Distanzmaße prüfen, wie beispielsweise der Cosinus-Ähnlichkeit oder darauf aufbauender Metriken wie dem BERTScore zur Bewertung von LLMs [8]. Auch der Einsatz fortgeschrittener Sprachmodelle oder die Integration domänenspezifischen Kontexts könnte zu einer signifikanten Steigerung der Extraktionsleistung beitragen. Darüber hinaus erscheint eine Evaluation des Ansatzes auf größeren und diversifizierten Datensätzen unter Berücksichtigung weiterer Ähnlichkeits- und Distanzmaße sinnvoll, um die Generalisierbarkeit der Ergebnisse zu überprüfen. Es liegt nahe zu vermuten, dass die Systemleistung durch zusätzliche Vorverarbeitungsschritte, etwa eine vorgelegte Dokumententyp-Klassifikation, weiter gesteigert werden könnte [34]. Auf dieser Basis könnten anschließend spezialisierte Prompts eingesetzt werden, die auf die jeweilige Dokumentenart abgestimmt sind.

Die im Rahmen dieser Arbeit erzielten Ergebnisse lassen sich prinzipiell durch verschiedene Maßnahmen erweitern. Zum einen besteht die Möglichkeit, das entwickelte System um zusätzliche Dokumententypen oder weitere Felder zu ergänzen, um eine breitere Abdeckung im Bereich zollrelevanter Unterlagen zu erreichen. Zum anderen kann die Extraktionspipeline durch gezielte und feinere Vorverarbeitungsschritte weiter verbessert werden. Darüber hinaus eröffnet der modulare Aufbau des Ansatzes Potenzial für die Einbindung zusätzlicher Verarbeitungsschritte, etwa zur Validierung, semantischen Anreicherung oder Nachbearbeitung der extrahierten Daten. Insgesamt bieten sich somit vielfältige Ansatzpunkte, um die Funktionalität und Leistungsfähigkeit des Systems sukzessive auszubauen und an sich wandelnde Anforderungen anzupassen.

Die Übertragbarkeit der im Rahmen dieser Arbeit gewonnenen Erkenntnisse auf andere Anwendungsbereiche oder Dokumententypen ist grundsätzlich gegeben, unterliegt jedoch bestimmten Einschränkungen. Insbesondere hängt der Erfolg einer Übertragung maßgeblich von der Ähnlichkeit der zugrundeliegenden Dokumentstrukturen sowie von der Verfügbarkeit passender Trainingsdaten ab. Für stark abweichende oder heterogene Dokumententypen kann eine erneute Anpassung der Extraktionspipeline oder eine gezielte Feinabstimmung der verwendeten Prompts erforderlich werden. Nichtsdestotrotz legt die entwickelte Methodik ein Fundament, das mit überschaubarem Aufwand für verwandte Aufgabenstellungen im Bereich der automatisierten Informationsextraktion adaptiert werden kann.

Literatur

- [1] Patrizio Bellan u. a. *Process Extraction from Text: Benchmarking the State of the Art and Paving the Way for Future Challenges*. arXiv:2110.03754 [cs]. Okt. 2023. DOI: 10.48550/arXiv.2110.03754. URL: <http://arxiv.org/abs/2110.03754> (besucht am 27.07.2025).
- [2] Michael Sherman u. a. „Reviewers and Contributors“. en. In: ().
- [3] Anisa Rula und Jennifer D’Souza. *Procedural Text Mining with Large Language Models*. arXiv:2310.03376 [cs]. Okt. 2023. DOI: 10.48550/arXiv.2310.03376. URL: <http://arxiv.org/abs/2310.03376> (besucht am 27.07.2025).
- [4] Noman Islam, Zeeshan Islam und Nazia Noor. *A Survey on Optical Character Recognition System*. arXiv:1710.05703 [cs]. Okt. 2017. DOI: 10.48550/arXiv.1710.05703. URL: <http://arxiv.org/abs/1710.05703> (besucht am 17.04.2025).
- [5] Jiawei Wang u. a. *Detect-Order-Construct: A Tree Construction based Approach for Hierarchical Document Structure Analysis*. arXiv:2401.11874 [cs]. März 2024. DOI: 10.48550/arXiv.2401.11874. URL: <http://arxiv.org/abs/2401.11874> (besucht am 02.06.2025).
- [6] Vishwanath D u. a. *Deep Reader: Information extraction from Document images via relation extraction and Natural Language*. arXiv:1812.04377 [cs]. Dez. 2018. DOI: 10.48550/arXiv.1812.04377. URL: <http://arxiv.org/abs/1812.04377> (besucht am 02.06.2025).
- [7] Kirill Smelyakov u. a. „Effectiveness of Modern Text Recognition Solutions and Tools for Common Data Sources“. en. In: ().
- [8] „(PDF) Performance Metrics for Multilabel Emotion Classification: Comparing Micro, Macro, and Weighted F1-Scores“. en. In: *ResearchGate* (Apr. 2025). DOI: 10.3390/app14219863. URL: https://www.researchgate.net/publication/385324843_Performance_Metrics_for_Multilabel_Emotion_Classification_Comparing_Micro_Macro_and_Weighted_F1-Scores (besucht am 28.07.2025).
- [9] Meharuniza Nazeem u. a. „Open-Source OCR Libraries: A Comprehensive Study for Low Resource Language“. In: *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*. Hrsg. von Sobha Lalitha Devi und Karunesh Arora. AU-KBC Research Centre, Chennai, In-

- dia: NLP Association of India (NLP AI), Dez. 2024, S. 416–421. URL: <https://aclanthology.org/2024.icon-1.48/> (besucht am 31.05.2025).
- [10] R. Smith. „An Overview of the Tesseract OCR Engine“. en. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*. ISSN: 1520-5363. Curitiba, Parana, Brazil: IEEE, Sep. 2007, S. 629–633. ISBN: 978-0-7695-2822-9. DOI: 10.1109/ICDAR.2007.4376991. URL: <http://ieeexplore.ieee.org/document/4376991/> (besucht am 17.04.2025).
- [11] Prakash Nadkarni, Lucila Ohno-Machado und Wendy Chapman. „Natural language processing: An introduction“. In: *Journal of the American Medical Informatics Association : JAMIA* 18 (Sep. 2011), S. 544–51. DOI: 10.1136/amiajnl-2011-000464.
- [12] Diksha Khurana u. a. *Natural Language Processing: State of The Art, Current Trends and Challenges*. en. Aug. 2017. DOI: 10.1007/s11042-022-13428-4. URL: <https://arxiv.org/abs/1708.05148v1> (besucht am 02.06.2025).
- [13] Matthew Honnibal u. a. „spaCy: Industrial-strength Natural Language Processing in Python“. In: (2020). DOI: 10.5281/zenodo.1212303.
- [14] Google. *Whitepaper Foundational Large Language Models & Text Generation*. eng. Okt. 2024. URL: <http://archive.org/details/whitepaper-foundational-large-language-models-text-generation> (besucht am 10.06.2025).
- [15] Imed Keraghel, Stanislas Morbieu und Mohamed Nadif. *Recent Advances in Named Entity Recognition: A Comprehensive Survey and Comparative Study*. en. Jan. 2024. URL: <https://arxiv.org/abs/2401.10825v3> (besucht am 03.06.2025).
- [16] Tong Xiao und Jingbo Zhu. *Foundations of Large Language Models*. arXiv:2501.09223 [cs]. Jan. 2025. DOI: 10.48550/arXiv.2501.09223. URL: <http://arxiv.org/abs/2501.09223> (besucht am 26.01.2025).
- [17] Ashish Vaswani u. a. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. DOI: 10.48550/arXiv.1706.03762. URL: <http://arxiv.org/abs/1706.03762> (besucht am 22.12.2024).
- [18] Stefania Cristina. *How to Implement Scaled Dot-Product Attention from Scratch in TensorFlow and Keras*. en-US. Sep. 2022. URL: <https://www.machinelearningmastery.com/how-to-implement-scaled-dot-product-attention-from-scratch-in-tensorflow-and-keras/> (besucht am 03.07.2025).
- [19] Jason Wei u. a. „Finetuned Language Models are Zero-Shot Learners“. In: Okt. 2021. URL: <https://openreview.net/forum?id=gEZrGCozdqR> (besucht am 19.02.2025).
- [20] OpenAI u. a. *GPT-4o System Card*. arXiv:2410.21276 [cs]. Okt. 2024. DOI: 10.48550/arXiv.2410.21276. URL: <http://arxiv.org/abs/2410.21276> (besucht am 15.04.2025).

- [21] Sakib Shahriar u. a. *Putting GPT-4o to the Sword: A Comprehensive Evaluation of Language, Vision, Speech, and Multimodal Proficiency*. arXiv:2407.09519 [cs]. Juni 2024. DOI: 10.48550/arXiv.2407.09519. URL: <http://arxiv.org/abs/2407.09519> (besucht am 15.04.2025).
- [22] Gavin Greif, Niclas Griesshaber und Robin Greif. *Multimodal LLMs for OCR, OCR Post-Correction, and Named Entity Recognition in Historical Documents*. arXiv:2504.00414 [cs]. Apr. 2025. DOI: 10.48550/arXiv.2504.00414. URL: <http://arxiv.org/abs/2504.00414> (besucht am 15.04.2025).
- [23] Gabriel Recchia. *Teaching Autoregressive Language Models Complex Tasks By Demonstration*. arXiv:2109.02102 [cs]. Dez. 2021. DOI: 10.48550/arXiv.2109.02102. URL: <http://arxiv.org/abs/2109.02102> (besucht am 19.02.2025).
- [24] Dhananjay Ashok und Zachary C. Lipton. *PromptNER: Prompting For Named Entity Recognition*. arXiv:2305.15444 [cs]. Juni 2023. DOI: 10.48550/arXiv.2305.15444. URL: <http://arxiv.org/abs/2305.15444> (besucht am 03.06.2025).
- [25] Yizhong Wang u. a. *Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks*. arXiv:2204.07705 [cs]. Okt. 2022. DOI: 10.48550/arXiv.2204.07705. URL: <http://arxiv.org/abs/2204.07705> (besucht am 27.06.2025).
- [26] Jiawei Chen u. a. *Learning In-context Learning for Named Entity Recognition*. arXiv:2305.11038 [cs]. Mai 2023. DOI: 10.48550/arXiv.2305.11038. URL: <http://arxiv.org/abs/2305.11038> (besucht am 27.06.2025).
- [27] Jiasheng Zhang u. a. „2INER: Instructive and In-Context Learning on Few-Shot Named Entity Recognition“. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Hrsg. von Houda Bouamor, Juan Pino und Kalika Bali. Singapore: Association for Computational Linguistics, Dez. 2023, S. 3940–3951. DOI: 10.18653/v1/2023.findings-emnlp.259. URL: <https://aclanthology.org/2023.findings-emnlp.259/> (besucht am 27.06.2025).
- [28] K. M. Sajjadul Islam u. a. *LLM-based Prompt Ensemble for Reliable Medical Entity Recognition from EHRs*. arXiv:2505.08704 [cs]. Mai 2025. DOI: 10.48550/arXiv.2505.08704. URL: <http://arxiv.org/abs/2505.08704> (besucht am 21.07.2025).
- [29] Andres Azqueta-Gavaldón und Joaquin Ramos Cosgrove. *Beyond Traditional Algorithms: Leveraging LLMs for Accurate Cross-Border Entity Identification*. arXiv:2507.11086 [cs]. Juli 2025. DOI: 10.48550/arXiv.2507.11086. URL: <http://arxiv.org/abs/2507.11086> (besucht am 21.07.2025).

- [30] Long Ouyang u. a. *Training language models to follow instructions with human feedback*. arXiv:2203.02155 [cs]. März 2022. DOI: 10.48550/arXiv.2203.02155. URL: <http://arxiv.org/abs/2203.02155> (besucht am 10.02.2025).
- [31] Andrew K. Lampinen u. a. *Can language models learn from explanations in context?* arXiv:2204.02329 [cs]. Okt. 2022. DOI: 10.48550/arXiv.2204.02329. URL: <http://arxiv.org/abs/2204.02329> (besucht am 19.02.2025).
- [32] Tom B. Brown u. a. *Language Models are Few-Shot Learners*. arXiv:2005.14165 [cs]. Juli 2020. DOI: 10.48550/arXiv.2005.14165. URL: <http://arxiv.org/abs/2005.14165> (besucht am 19.02.2025).
- [33] Zeyi Wen u. a. „2ED: An Efficient Entity Extraction Algorithm Using Two-Level Edit-Distance“. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. ISSN: 2375-026X. Apr. 2019, S. 998–1009. DOI: 10.1109/ICDE.2019.00093. URL: <https://ieeexplore.ieee.org/document/8731344/> (besucht am 27.07.2025).
- [34] Shuo Jiang u. a. „Deep Learning for Technical Document Classification“. In: *IEEE Transactions on Engineering Management* 71 (2024). arXiv:2106.14269 [cs], S. 1163–1179. ISSN: 0018-9391, 1558-0040. DOI: 10.1109/TEM.2022.3152216. URL: <http://arxiv.org/abs/2106.14269> (besucht am 28.07.2025).

A. Anhang A

A.1. Implementierung der Extraktionspipeline

In Listing A.1 ist die Umsetzung der Umwandlung der PDF-Dateien zu Bildformaten sowie der Extraktionsschritt über das LLM.

```
1  # Module
2  import dataiku
3  import pandas as pd, numpy as np
4  from dataiku import pandasutils as pdu
5  import os
6  import fitz # PyMuPDF
7  import io
8  from PIL import Image
9  import base64
10 import json
11 from openai import AzureOpenAI
12 import tiktoken
13 import logging
14 import re
15
16
17 logger = logging.getLogger(__name__)
18
19
20 logging.basicConfig(
21     level=logging.INFO,
22     format='%(asctime)s [%(levelname)s] [%(name)s] %(message)s'
23 )
24
25
26 # Daten einlesen
27 zolldokumente = dataiku.Folder("8hG3cHjI")
28 prompts = dataiku.Dataset("Prompts_Task_1")
29 prompts_df = prompts.get_dataframe()
30
31 # Projektvariablen für Azure OpenAI einlesen
32 project = dataiku.Project()
33 variables = project.get_variables()
34
```

```

35 AZURE_OPENAI_ENDPOINT =
    variables['standard']['AZURE_OPENAI_ENDPOINT']
36 AZURE_OPENAI_KEY = variables['standard']['AZURE_OPENAI_KEY']
37 AZURE_OPENAI_API_VERSION =
    variables['standard']['AZURE_OPENAI_API_VERSION']
38 AZURE_OPENAI_MODEL = variables['standard']['AZURE_OPENAI_MODEL']
39 FOLDER_URL = variables['standard']['FOLDER_URL']
40
41 # Tiktoken um Tokenanzahl pro Dokument zu bestimmen
42 encoding = tiktoken.encoding_for_model('gpt-4o')
43
44
45 # Azure OpenAI client
46 client = AzureOpenAI(
47     azure_endpoint=AZURE_OPENAI_ENDPOINT,
48     api_key=AZURE_OPENAI_KEY,
49     api_version=AZURE_OPENAI_API_VERSION
50 )
51
52 # Prompts der verschiedenen Verfahren
53 parameters = prompts_df[['Version', 'System Prompt', 'User
    Prompt']]
54
55
56 # Funktion um PDF in Bild umzuwandeln
57 def pdf_to_base64_images(pdf_stream):
58     try:
59         pdf_document = fitz.open(stream=pdf_stream,
60                                 filetype="pdf") # Opening the PDF from the stream
61         base64_images = []
62
63         for page_num in range(len(pdf_document)):
64             logger.info(f"[PDF->BASE64]: pagenumber {page_num}")
65             page = pdf_document.load_page(page_num) # Get the
66                 page
67             pix = page.get_pixmap() # Convert the page to a
68                 pixmap (image)
69             img = Image.open(io.BytesIO(pix.tobytes())) #
70                 Convert pixmap to PIL Image
71
72             # Save image to BytesIO instead of temporary file
73             img_byte_arr = io.BytesIO()
74             img.save(img_byte_arr, format="PNG")
75             img_byte_arr.seek(0)
76
77             # Convert the image to base64
78             base64_image =
79                 base64.b64encode(img_byte_arr.read()).decode('utf-8')
80             base64_images.append(base64_image)

```

```

76
77     return base64_images
78 except Exception as e:
79     logger.error(f"[PDF TO BASE 64] Error: {e}")
80
81
82
83 # Funktion für Extraktion mit OpenAI Gpt-4o
84 def extract_customs_data(base64_image, url, system_prompt,
85     user_prompt, max_retries=5, backoff_factor=2):
86     last_exception = None
87     for attempt in range(1, max_retries + 1):
88         try:
89             logger.info(f"[GPT4 CALL]: sys prompt
90                 {system_prompt[:20]}")
91             logger.info(f"[PDF->BASE64]: usr prompt
92                 {user_prompt[:20]}")
93
94             # LLM Aufruf
95             response = client.chat.completions.create(
96                 model=AZURE_OPENAI_MODEL,
97                 response_format={ "type": "json_object" },
98                 messages=[
99                     {
100                         "role": "system",
101                         "content": system_prompt.format(url=url)
102                     },
103                     {
104                         "role": "user",
105                         "content": [
106                             {"type": "text", "text": user_prompt},
107                             {"type": "image_url", "image_url":
108                                 {"url":
109                                     f"data:image/png;base64,{base64_image}",
110                                     "detail": "high"}}
111                         ]
112                     }
113                 ],
114                 temperature=0.0,
115             )
116             return response.choices[0].message.content
117 except Exception as e:
118     last_exception = e
119     logger.warning(f"[GPT4 EXTRACT] Attempt
120         {attempt}/{max_retries} failed: {e}")
121     if attempt < max_retries:
122         sleep_time = backoff_factor ** (attempt - 1)
123         logger.info(f"[GPT4 EXTRACT]Retrying in
124             {sleep_time} seconds...")

```

```

117         time.sleep(sleep_time)
118     logger.error(f"[GPT4 EXTRACT] All {max_retries} attempts
119         failed.")
120     raise last_exception
121
122
123 # Funktion für Multiseiten PDF extraktion und Umwandlung in JSON
124 def extract_from_multiple_pages(base64_images, original_filename,
125     sendungsnummer, output_folder, url, system_prompt,
126     user_prompt, version):
127     entire_documents = []
128     tokens_count = {'count_tokens_doc':0}
129     version_prompt = {'prompt_version':version}
130     sendung = {'sendung': sendungsnummer}
131
132     for num, base64_image in enumerate(base64_images):
133         try:
134             document_json = extract_customs_data(base64_image,
135                 url, system_prompt, user_prompt)
136             document_data = json.loads(document_json)
137             logger.info(f"[DOC EXTRACT MULT] doc number: {num},
138                 version: {version}")
139             logger.info(f"[DOC EXTRACT MULT] doc content:
140                 {document_json[:200]}")
141             if isinstance(document_data, list):
142                 entire_documents.extend(document_data)
143             elif isinstance(document_data, dict):
144                 entire_documents.append(document_data)
145             else:
146                 logger.warning(f"Unerwarteter Typ von
147                     document_data: {type(document_data)}")
148         except Exception as e:
149             logger.error(f"[GPT4 EXTRACT MULT] Error: {e}")
150             tokens = encoding.encode(str(entire_documents))
151             tokens_count['count_tokens_doc'] = len(tokens)
152
153     entire_documents.append(sendung)
154     entire_documents.append(tokens_count)
155     entire_documents.append(version_prompt)
156
157     # Dateiname definieren
158     output_filename = original_filename.replace('.pdf',
159         f'extracted_{version}.json')
160
161     # Json in verteiltes Dateisystem schreiben
162     json_object = json.dumps(entire_documents,
163         ensure_ascii=False, indent=4).encode('utf-8')
164     with output_folder.get_writer(output_filename) as writer:

```

```
157         writer.write(json_object)
158
159 # Funktionsaufrufe
160 output_folder = dataiku.Folder()
161 for version, system_prompt, user_prompt in
    parameters.itertuples(index=False, name=None):
162     for path in zolldokumente.list_paths_in_partition():
163         sendungsnummer = path.split('/')[1].split('/')[0]
164         url = FOLDER_URL + path
165         filename = os.path.basename(path)
166         if path.endswith(".pdf"):
167             try:
168                 # PDF zu Bild konvertieren
169                 with zolldokumente.get_download_stream(path) as
                    stream:
170                     pdf_bytes = stream.read()
171                     pdf_stream = io.BytesIO(pdf_bytes)
172                     base64_images =
                        pdf_to_base64_images(pdf_stream)
173
174                 # Dateiinhalt extrahieren
175                 extract_from_multiple_pages(base64_images,
                    filename, sendungsnummer, output_folder, url,
                    system_prompt, user_prompt, version)
176             except Exception as e:
177                 logger.error(f"Error processing {path}: {e}")
```

Listing A.1: LLM als OCR Ersatz - Umwandlung von PDF in Bild und Extraktion der Entitäten von Bild zu Text

Listing A.2 beschreibt die Implementierung der Schema-Migration über das LLM.

```
1  # Module
2  import dataiku
3  import pandas as pd, numpy as np
4  from dataiku import pandasutils as pdu
5  import os
6  import io
7  import json
8  from openai import AzureOpenAI
9  from pydantic import BaseModel
10 from typing import List, Optional
11 from datetime import datetime
12
13 # Pydantic Modelle um Schema Prüfung für Output durchzuführen
14 class Zollkopfdaten(BaseModel):
15     frachtbrief_packstueckanzahl: Optional[int]
16     frachtbrief_bruttogewicht: Optional[float]
17     frachtbrief_versendungsdatum: Optional[datetime]
18     zolldokument_MRN: Optional[str]
19     zolldokument_verschluss_feld_19_10: Optional[str]
20     zolldokument_wareneingangsnummer_stempel: Optional[str]
21     zolldokument_frist: Optional[datetime]
22     zolldokument_positionen_insgesamt: Optional[int]
23     zolldokument_packstücke_insgesamt: Optional[int]
24     zolldokument_gesamtrohmasse: Optional[float]
25     rechnung_rechnungsnummer: Optional[str]
26     rechnung_rechnungsdatum: Optional[datetime]
27     rechnung_bestellnummern: List[Optional[str]]
28     rechnung_lieferplan: Optional[str]
29     rechnung_einzelpositionen_gesamtanzahl: Optional[int]
30     rechnung_ursprungserklaerung_text: Optional[str]
31     lieferschein_lieferscheinnummern: List[Optional[str]]
32     lieferschein_lieferantennamen: Optional[str]
33     lieferschein_versenderland: Optional[str]
34     lieferschein_lieferplan: Optional[str]
35     lieferschein_einzelpositionen_anzahl: Optional[int]
36     lieferschein_ursprungserklaerung_text: Optional[str]
37     incoterm: Optional[str]
38     eori_gb_ukraine_registrierungsnummer: Optional[str]
39     atr_feld_4: Optional[str]
40     atr_feld_5: Optional[str]
41     atr_feld_6: Optional[str]
42     atr_feld_12_stempel: Optional[str]
43     atr_feld_13_stempel: Optional[str]
44     eurl_feld_2: Optional[str]
45     eurl_feld_4: Optional[str]
46     eurl_feld_5: Optional[str]
47     eurl_feld_11_stempel: Optional[str]
48     eurl_feld_12_stempel: Optional[str]
```

```
49     sendungsnummer: None
50     referenz_zu_dokument: str
51
52
53 class Zolleinzelpositionen(BaseModel):
54     artikelnummer: Optional[str]
55     preis: Optional[float]
56     waehrung: Optional[str]
57     menge: Optional[float]
58     menge_maßeinheit: Optional[str]
59     eigenmasse: Optional[float]
60     eigenmasse_maßeinheit: Optional[str]
61     anmelde_und_handelsstatistische_menge: Optional[float]
62     beantragte_beguenstigung: Optional[str]
63     betriebliche_identifikationsnummer: Optional[str]
64     container_nummer: Optional[str]
65     packstueckzeichen_und_nummer: Optional[str]
66     packstueckanzahl: Optional[int]
67     packstueckart: Optional[str]
68     rohmasse: Optional[float]
69     rohmasse_maßeinheit: Optional[str]
70     ursprungsland: Optional[str]
71     warenbezeichnung: Optional[str]
72     warennummer: Optional[str]
73     sendungsnummer: None
74     referenz_zu_dokument: str
75
76
77 class ZollSchema(BaseModel):
78     document_data: Zollkopfdaten
79     goods_positions: List[Zolleinzelpositionen]
80     sendung: str
81     token_count_doc: int
82     prompt_version: str
83
84
85 # Dateien einlesen
86 json_ohne_schema = dataiku.Folder()
87 prompts = dataiku.Dataset()
88 prompts_df = prompts.get_dataframe()
89 schema = dataiku.Folder()
90
91 # Projekt Variablen für Azure OpenAI
92 project = dataiku.Project()
93 variables = project.get_variables()
94 AZURE_OPENAI_ENDPOINT =
95     variables['standard']['AZURE_OPENAI_ENDPOINT']
96 AZURE_OPENAI_KEY = variables['standard']['AZURE_OPENAI_KEY']
97 AZURE_OPENAI_API_VERSION =
```



```

141         }
142     ]
143 }
144 ],
145     response_format=ZollSchema,
146     temperature=0.0,
147 )
148 return response.choices[0].message.parsed
149
150 output_folder = dataiku.Folder()
151
152 for version, system_prompt, user_prompt in
    parameters.itertuples(index=False, name=None):
153     for path in json_ohne_schema.list_paths_in_partition():
154         # Dateiname definieren
155         filename = os.path.basename(path)
156         if path.endswith(".json"):
157             try:
158                 with json_ohne_schema.get_download_stream(path)
159                     as stream:
160                     json_objects = json.load(stream)
161
162                     tokens = [obj.get('tokens', 0) for obj in
163                             json_objects]
164                     total_token_count = sum(tokens)
165
166                     # Transformieren der Json Datei in vordefiniertes
167                     # Schema
168                     transformed_json =
169                         transform_customs_data(json_objects,
170                                                 system_prompt, user_prompt, total_token_count)
171                     json_string =
172                         transformed_json.model_dump_json(indent=1)
173                     transformed_filename = f"transformed_{filename}"
174                     with
175                         output_folder.get_writer(transformed_filename)
176                         as writer:
177                         writer.write(json_string.encode("utf-8"))
178             except Exception as e:
179                 print(f"Error processing {path}: {e}")

```

Listing A.2: LLM um Daten in vordefiniertes Schema zu migrieren

A.2. Prompt Design für Datenextraktion

Ergänzend zu dem definierten Prompt aus ?? werden im Folgenden die übrigen Prompting-Verfahren vorgestellt, um die Leistungsfähigkeit der Informationsextraktion weiter zu erhöhen. Dabei werden One-Shot Prompting und Few-Shot Prompting jeweils mit instruktionsbasiertem Prompting kombiniert.

- **One-Shot- mit instruktionsbasiertem Prompting**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and transport documents. There are six main types of documents:

- Waybill (Frachtbrief)
- T1 Transit Document (Zolldokument)
- Invoice (Rechnung)
- Delivery Note (Lieferschein)
- Movement Certificate (ATR)
- Movement Certificate (EUR1)

You are an OCR-like data extraction tool specialized in customs and transport documents. Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making up data if not available write `None` as value. Use the one given example and generalize it to solve your task, also follow the given instructions step by step to solve your task. Always provide the key `"referenz zu dokument": url`.

User Prompt:

Here are the instructions:

1. Extract all relevant fields of the document.
2. Keep values in original language.
3. The reference of the document is: {url}.

Here is the example:

Example:

Text: "Invoice No: INV-0098, Date: 2024-05-12, Sender: ABC GmbH, Country: Germany"

JSON:

```

{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    ...
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma.sharepoint.com/...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_maßeinheit": "Stück",
      "eigenmasse": None,
      ...
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
      "warennummer": "73049051",
      "dateiname": "Position_1.pdf",
      "sendungsnummer": "SN-20240628-001",
      "referenz_zu_dokument": "https://firma...pdf"
    },
    "... more article if applicable ..."
  ]
}

```

Extract all relevant fields from the document.

- **Kombination Few-Shot- mit instuktionsbasiertem Prompting für Datenextraktion**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and

transport documents. There are six main types of documents:

- Waybill (Frachtbrief)
- T1 Transit Document (Zolldokument)
- Invoice (Rechnung)
- Delivery Note (Lieferschein)
- Movement Certificate (ATR)
- Movement Certificate (EUR1)

You are an OCR-like data extraction tool specialized in customs and transport documents. Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making up data if not available write `None` as value. Use the one given example and generalize it to solve your task, also follow the given instructions step by step to solve your task. Always provide the key `"referenz zu dokument": url`.

User Prompt:

Here are the instructions:

1. Extract all relevant fields of the document.
2. Keep values in original language.
3. The reference of the document is: {url}.

Here is the example:

Example:

Text: Invoice No: INV-0098, Date: 2024-05-12, Sender: ABC GmbH, Country: Germany"

Example 1 – JSON Output:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
```

```

    "referenz_zu_dokument": "https://firma...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_maßeinheit": "Stück",
      "eigenmasse": None,
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
      "warennummer": "73049051",
      "dateiname": "Position_1.pdf",
      "sendungsnummer": "SN-20240628-001",
      "referenz_zu_dokument": "https://firma...pdf"
    }
  ]
}

```

Example 2: Text: Shipment number: SN-20240701-002

Waybill information: Package count: 8 Gross weight: 4550.0 kg Shipping date: 2024-07-01 14:00

Customs document: MRN: 20DE9876543210ZYXW2 Seal (field 19 10): SGL2025

... omitted fields ...

EUR.1 certificate: Field 4: Deutschland Field 5: Italien Field 11 stamp: (not specified) Field 12 stamp: (not specified)

Document reference: https://company....pdf

Goods positions: 1. Article number: B-2047 Price: 320.40 USD Quantity: 5 Box Net mass: 120.5 kg Country of origin: USA Goods description: Electrical Components Commodity code: 85423910 File name: Position 2.pdf Shipment number: SN-20240701-002 Reference document: https://company...pdf

... more article if applicable ...

Example 2 – JSON Output:

```
{
```

```

"document_data": {
  "sendungsnummer": "SN-20240701-002",
  "frachtbrief_packstueckanzahl": 8,
  "frachtbrief_bruttogewicht": 4550.0,
  "frachtbrief_versendungsdatum": "2024-07-01T14:00:00",
  "zolldokument_MRN": "20DE9876543210ZYXW2",
  "zolldokument_verschluss_feld_19_10": "SGL2025",
  "eurl_feld_4": "Deutschland",
  "eurl_feld_5": "Italien",
  "eurl_feld_11_stempel": None,
  "eurl_feld_12_stempel": None,
  "referenz_zu_dokument": "https://company...pdf"
},
"goods_positions": [
  {
    "artikelnummer": "B-2047",
    "preis": 320.40,
    "währung": "USD",
    "menge": 5,
    "menge_masseinheit": "Box",
    "eigenmasse": 120.5,
    "ursprungsland": "USA",
    "warenbezeichnung": "Electrical Components",
    "warennummer": "85423910",
    "dateiname": "Position_2.pdf",
    "sendungsnummer": "SN-20240701-002",
    "referenz_zu_dokument": "https://company...pdf"
  }
]
}

```

Extract all relevant fields from the document.

- **One-Shot Prompting**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and transport documents. There are six main types of documents: - Waybill (Frachtbrief) - T1 Transit Document (Zolldokument) - Invoice (Rechnung) - Delivery Note (Lieferschein) - Movement Certificate (ATR) - Movement Certificate (EUR1)

Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making up data if not available write `None` as value. Use the one given example and generalize it to solve your task. Always provide the key `"referenz zu dokument": {url}`.

User Prompt:

Here is the example:

Example:

Text: Invoice No: INV-0098, Date: 2024-05-12, Sender: ABC GmbH, Country: Germany"

JSON:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    ...
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma.sharepoint.com/...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_masseinheit": "Stück",
      "eigenmasse": None,
      ...
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
      "warennummer": "73049051",
      "dateiname": "Position_1.pdf",
      "sendungsnummer": "SN-20240628-001",
    }
  ]
}
```

```

        "referenz_zu_dokument": "https://firma...pdf"
    },
    "... more article if applicable ..."
]
}

```

Extract all relevant fields from the document.

- **Few-Shot Prompting**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and transport documents. There are six main types of documents: - Waybill (Frachtbrief) - T1 Transit Document (Zolldokument) - Invoice (Rechnung) - Delivery Note (Lieferschein) - Movement Certificate (ATR) - Movement Certificate (EUR1)

Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making up data if not available write `None` as value. Use the given examples and generalize them to solve your task. Always provide the key `"referenz_zu_dokument": {url}`.

User Prompt:

Example 1: Text: Shipment number: SN-20240628-001

Waybill information: Package count: (not specified) Gross weight: 7850.5 kg Shipping date: (not specified)

Customs document: MRN: 19DE1234567890ABCDE1 Seal (field 19 10): (not specified)

... omitted fields ...

EUR.1 certificate: Field 4: (not specified) Field 5: France Field 11 stamp: (not specified) Field 12 stamp: StampEUR1CustomsB

Document reference: "https://firma...pdf"

Goods positions: 1. Article number: A-1023 Price: 1590.75 EUR Quantity: 10 pieces Net mass: (not specified) Country of origin: Germany Goods description: Steel tube Commodity code: 73049051 File name: Position 1.pdf Shipment number: SN-20240628-001

... omitted ...

Example 1 – JSON Output:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_masseinheit": "Stück",
      "eigenmasse": None,
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
      "warennummer": "73049051",
      "dateiname": "Position_1.pdf",
      "sendungsnummer": "SN-20240628-001",
      "referenz_zu_dokument": "https://firma...pdf"
    }
  ]
}
```

Example 2: Text: Shipment number: SN-20240701-002

Waybill information: Package count: 8 Gross weight: 4550.0 kg Shipping date: 2024-07-01 14:00

Customs document: MRN: 20DE9876543210ZYXW2 Seal (field 19 10): SGL2025

... omitted fields ...

EUR.1 certificate: Field 4: Deutschland Field 5: Italien Field 11 stamp: (not specified) Field 12 stamp: (not specified)

Document reference: <https://company....pdf>

Goods positions: 1. Article number: B-2047 Price: 320.40 USD Quantity: 5 Box Net mass: 120.5 kg Country of origin: USA Goods description: Electrical Components Commodity code: 85423910 File name: Position 2.pdf Shipment number: SN-20240701-002 Reference document: <https://company...pdf>

... more article if applicable ...

Example 2 – JSON Output:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240701-002",
    "frachtbrief_packstueckanzahl": 8,
    "frachtbrief_bruttogewicht": 4550.0,
    "frachtbrief_versendungsdatum": "2024-07-01T14:00:00",
    "zolldokument_MRN": "20DE9876543210ZYXW2",
    "zolldokument_verschluss_feld_19_10": "SGL2025",
    "eurl_feld_4": "Deutschland",
    "eurl_feld_5": "Italien",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": None,
    "referenz_zu_dokument": "https://company...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "B-2047",
      "preis": 320.40,
      "währung": "USD",
      "menge": 5,
      "menge_masseinheit": "Box",
      "eigenmasse": 120.5,
      "ursprungsland": "USA",
      "warenbezeichnung": "Electrical Components",
      "warennummer": "85423910",
      "dateiname": "Position_2.pdf",
      "sendungsnummer": "SN-20240701-002",
      "referenz_zu_dokument": "https://company...pdf"
    }
  ]
}
```

```
}
```

Extract all relevant fields from the document.

- **Instruktionsbasiertes Prompting**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and transport documents. There are six main types of documents: - Waybill (Frachtbrief) - T1 Transit Document (Zolldokument) - Invoice (Rechnung) - Delivery Note (Lieferschein) - Movement Certificate (ATR) - Movement Certificate (EUR1) You are an OCR-like data extraction tool specialized in customs and transport documents. Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making the up data if not available write None as value. Follow the given instructions step by step to solve your task. Always provide the key "referenz zu dokument: url

User Prompt:

Here are the instructions: 1. Extract all relevant fields of the document. 2. Keep values in original language. 3. The reference of the document is: url.

A.3. Prompt Design für Schemamigration

Die verschiedenen Promptingstrategien wurden ebenso für die Schemamigration eingesetzt und evaluiert. Dabei dienen die in ?? beschriebenen Prompttechniken ebenso als Grundlage, um strukturierte JSON-Outputs gemäß einem vorab definierten Zieldatenschema zu generieren. Dies erlaubt eine automatische Überführung semistrukturierter Dokumenteninhalte in strukturierte Datenrepräsentationen, welche für die Weiterverarbeitung in nachgelagerten IT-Systemen genutzt werden können.

- **One-Shot Prompting**

System Prompt:

You are a data transformation tool that takes in JSON data and a reference JSON schema, and outputs JSON data according to the schema. Not all of the data in the input JSON will fit the schema, so you may need to omit some data or add null values to the output JSON. Translate all data into German if not already in German. Ensure values are formatted as specified in the schema (e.g. dates as YYYY-MM-DD). Here is the schema: {schema json}. Do the task by taking the example into account.

User Prompt:

Here is the example:

Example:

Text: Invoice No: INV-0098, Date: 2024-05-12, Sender: ABC GmbH,
Country: Germany"

JSON:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    ...
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma.sharepoint.com/...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_masseinheit": "Stück",
      "eigenmasse": None,
      ...
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
      "warennummer": "73049051",
    }
  ]
}
```

```

        "dateiname": "Position_1.pdf",
        "sendungsnummer": "SN-20240628-001",
        "referenz_zu_dokument": "https://firma...pdf"
    },
    "... more article if applicable ..."
]
}

```

Transform the following raw JSON data according to the provided schema. Ensure all data is in English and formatted as specified by values in the schema. Here is the raw JSON: {json raw}.

- **Few-Shot Prompting**

System Prompt:

You are a data transformation tool that takes in JSON data and a reference JSON schema, and outputs JSON data according to the schema. Not all of the data in the input JSON will fit the schema, so you may need to omit some data or add null values to the output JSON. Translate all data into German if not already in German. Ensure values are formatted as specified in the schema (e.g. dates as YYYY-MM-DD). Here is the schema: {schema json}. Do the task by taking the examples into account.

User Prompt:

Example 1: Text: Shipment number: SN-20240628-001

Waybill information: Package count: (not specified) Gross weight: 7850.5 kg Shipping date: (not specified)

Customs document: MRN: 19DE1234567890ABCDE1 Seal (field 19 10): (not specified)

... omitted fields ...

EUR.1 certificate: Field 4: (not specified) Field 5: France Field 11 stamp: (not specified) Field 12 stamp: StampEUR1CustomsB

Document reference: "https://firma...pdf"

Goods positions: 1. Article number: A-1023 Price: 1590.75 EUR Quantity: 10 pieces Net mass: (not specified) Country of origin: Germany Goods description: Steel tube Commodity code: 73049051 File name:

Position 1.pdf Shipment number: SN-20240628-001

... omitted ...

Example 1 – JSON Output:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_masseinheit": "Stück",
      "eigenmasse": None,
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
      "warennummer": "73049051",
      "dateiname": "Position_1.pdf",
      "sendungsnummer": "SN-20240628-001",
      "referenz_zu_dokument": "https://firma...pdf"
    }
  ]
}
```

Example 2: Text: Shipment number: SN-20240701-002

Waybill information: Package count: 8 Gross weight: 4550.0 kg Shipping date: 2024-07-01 14:00

Customs document: MRN: 20DE9876543210ZYXW2 Seal (field 19 10): SGL2025

... omitted fields ...

EUR.1 certificate: Field 4: Deutschland Field 5: Italien Field 11 stamp:
(not specified) Field 12 stamp: (not specified)

Document reference: <https://company....pdf>

Goods positions: 1. Article number: B-2047 Price: 320.40 USD Quantity: 5 Box Net mass: 120.5 kg Country of origin: USA Goods description: Electrical Components Commodity code: 85423910 File name: Position 2.pdf Shipment number: SN-20240701-002 Reference document: <https://company...pdf>

... more article if applicable ...

Example 2 – JSON Output:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240701-002",
    "frachtbrief_packstueckanzahl": 8,
    "frachtbrief_bruttogewicht": 4550.0,
    "frachtbrief_versendungsdatum": "2024-07-01T14:00:00",
    "zolldokument_MRN": "20DE9876543210ZYXW2",
    "zolldokument_verschluss_feld_19_10": "SGL2025",
    "eurl_feld_4": "Deutschland",
    "eurl_feld_5": "Italien",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": None,
    "referenz_zu_dokument": "https://company...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "B-2047",
      "preis": 320.40,
      "währung": "USD",
      "menge": 5,
      "menge_masseinheit": "Box",
      "eigenmasse": 120.5,
      "ursprungsland": "USA",
      "warenbezeichnung": "Electrical Components",
      "warennummer": "85423910",
      "dateiname": "Position_2.pdf",
      "sendungsnummer": "SN-20240701-002",
      "referenz_zu_dokument": "https://company...pdf"
    }
  ]
}
```

```

    }
  ]
}

```

Transform the following raw JSON data according to the provided schema. Ensure all data is in German and formatted as specified by values in the schema. Here is the raw JSON: {json raw}.

- **Instruktionsbasiertes Prompting**

System Prompt:

You are a data transformation tool that takes in JSON data and a reference JSON schema, and outputs JSON data according to the schema. Not all of the data in the input JSON will fit the schema, so you may need to omit some data or add null values to the output JSON. Translate all data into German if not already in German. Ensure values are formatted as specified in the schema (e.g. dates as YYYY-MM-DD). Here is the schema: {schema json}. Do the task by following the given instructions.

User Prompt:

Here are the instructions:

1. Transform the following raw JSON data according to the provided schema.
2. Assign the values to the corresponding new key if applicable; otherwise assign null values to the key.
3. Ensure all data is in German and formatted as specified by values in the schema.
4. Rather include empty fields as null, than interpolate or make them up.
5. Here is the raw JSON: {json raw}.

- **One-Shot und Instruktionsbasiertes Prompting**

System Prompt:

You are a data transformation tool that takes in JSON data and a reference JSON schema, and outputs JSON data according to the schema. Not all of the data in the input JSON will fit the schema, so you may need to omit some data or add null values to the output

JSON. Translate all data into German if not already in German. Ensure values are formatted as specified in the schema (e.g. dates as YYYY-MM-DD). Here is the schema: {schema json}. Do the task by following the given instructions and by taking the example into account.

User Prompt:

Here are the instructions:

1. Transform the following raw JSON data according to the provided schema.
2. Assign the values to the corresponding new key if applicable; otherwise assign null values to the key.
3. Ensure all data is in German and formatted as specified by values in the schema.
4. Rather include empty fields as null, than interpolate or make them up.
5. Here is the raw JSON: {json raw}.

Here is the example:

Text: Invoice No: INV-0098, Date: 2024-05-12, Sender: ABC GmbH, Country: Germany"

JSON:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    ...
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma.sharepoint.com/...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
```

```

    "preis": 1590.75,
    "währung": "EUR",
    "menge": 10,
    "menge_máßeinheit": "Stück",
    "eigenmasse": None,
    ...
    "ursprungsland": "Deutschland",
    "warenbezeichnung": "Stahlrohr",
    "warennummer": "73049051",
    "dateiname": "Position_1.pdf",
    "sendungsnummer": "SN-20240628-001",
    "referenz_zu_dokument": "https://firma...pdf"
  },
  "... more article if applicable ..."
]
}

```

Now extract the relevant fields.

- **Few-Shot- mit instuktionsbasiertem Prompting für Datenextraktion**

System Prompt:

You are an OCR-like data extraction tool specialized in customs and transport documents. There are six main types of documents:

- Waybill (Frachtbrief)
- T1 Transit Document (Zolldokument)
- Invoice (Rechnung)
- Delivery Note (Lieferschein)
- Movement Certificate (ATR)
- Movement Certificate (EUR1)

You are an OCR-like data extraction tool specialized in customs and transport documents. Your task is to identify the type and extract structured data in JSON format from invoices, delivery notes, customs documents, and freight forms. Follow the structure and precision expected in official customs workflows. Instead of making up data if not available write `None` as value. Use the one given example and generalize it to solve your task, also follow the given instructions step by step to solve your task. Always provide the key `"referenz zu dokument": url`.

User Prompt:

Here are the instructions:

1. Transform the following raw JSON data according to the provided schema.
2. Assign the values to the corresponding new key if applicable; otherwise assign null values to the key.
3. Ensure all data is in German and formatted as specified by values in the schema.
4. Rather include empty fields as null, than interpolate or make them up.
5. Here is the raw JSON: json_raw.

Here is the example:

Text: Invoice No: INV-0098, Date: 2024-05-12, Sender: ABC GmbH, Country: Germany"

Example 1 – JSON Output:

```
{
  "document_data": {
    "sendungsnummer": "SN-20240628-001",
    "frachtbrief_packstueckanzahl": None,
    "frachtbrief_bruttogewicht": 7850.5,
    "frachtbrief_versendungsdatum": None,
    "zolldokument_MRN": "19DE1234567890ABCDE1",
    "zolldokument_verschluss_feld_19_10": None,
    "eurl_feld_4": None,
    "eurl_feld_5": "Frankreich",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": "StempelEUR1ZollB",
    "referenz_zu_dokument": "https://firma...pdf"
  },
  "goods_positions": [
    {
      "artikelnummer": "A-1023",
      "preis": 1590.75,
      "währung": "EUR",
      "menge": 10,
      "menge_máßeinheit": "Stück",
      "eigenmasse": None,
      "ursprungsland": "Deutschland",
      "warenbezeichnung": "Stahlrohr",
    }
  ]
}
```

```

        "warennummer": "73049051",
        "dateiname": "Position_1.pdf",
        "sendungsnummer": "SN-20240628-001",
        "referenz_zu_dokument": "https://firma...pdf"
    }
]
}

```

Example 2: Text: Shipment number: SN-20240701-002

Waybill information: Package count: 8 Gross weight: 4550.0 kg Shipping date: 2024-07-01 14:00

Customs document: MRN: 20DE9876543210ZYXW2 Seal (field 19 10): SGL2025

... omitted fields ...

EUR.1 certificate: Field 4: Deutschland Field 5: Italien Field 11 stamp: (not specified) Field 12 stamp: (not specified)

Document reference: <https://company....pdf>

Goods positions: 1. Article number: B-2047 Price: 320.40 USD Quantity: 5 Box Net mass: 120.5 kg Country of origin: USA Goods description: Electrical Components Commodity code: 85423910 File name: Position 2.pdf Shipment number: SN-20240701-002 Reference document: <https://company...pdf>

... more article if applicable ...

Example 2 – JSON Output:

```

{
  "document_data": {
    "sendungsnummer": "SN-20240701-002",
    "frachtbrief_packstueckanzahl": 8,
    "frachtbrief_bruttogewicht": 4550.0,
    "frachtbrief_versendungsdatum": "2024-07-01T14:00:00",
    "zolldokument_MRN": "20DE9876543210ZYXW2",
    "zolldokument_verschluss_feld_19_10": "SGL2025",
    "eurl_feld_4": "Deutschland",
    "eurl_feld_5": "Italien",
    "eurl_feld_11_stempel": None,
    "eurl_feld_12_stempel": None,
    "referenz_zu_dokument": "https://company...pdf"
  }
}

```

```
    },
    "goods_positions": [
      {
        "artikelnummer": "B-2047",
        "preis": 320.40,
        "währung": "USD",
        "menge": 5,
        "menge_masseinheit": "Box",
        "eigenmasse": 120.5,
        "ursprungsland": "USA",
        "warenbezeichnung": "Electrical Components",
        "warennummer": "85423910",
        "dateiname": "Position_2.pdf",
        "sendungsnummer": "SN-20240701-002",
        "referenz_zu_dokument": "https://company...pdf"
      }
    ]
  }
```

Now extract the relevant fields.

B. Anhang B

Ergänzend zu Kapitel 5 die Ergebnisse auf Klassenebene.

Prompttyp	Klasse	TP	FP	FN	Accuracy	Precision	Recall	F1	Support	Gpt-4o
Zero-Shot	rechnung_rechnungsnummer	11	5	12	0,39	0,69	0,48	0,56	23	16
Zero-Shot	lieferschein_lieferantennamen	4	8	5	0,24	0,33	0,44	0,38	9	12
Zero-Shot	lieferschein_versenderland	4	8	5	0,24	0,33	0,44	0,38	9	12
Zero-Shot	incoterm	4	8	9	0,19	0,33	0,31	0,32	13	12
Zero-Shot	frachtbrief_bruttogewicht	4	4	11	0,21	0,5	0,27	0,35	15	8
Zero-Shot	lieferschein_lieferscheinnummern	6	23	23	0,12	0,21	0,21	0,21	29	29
Zero-Shot	rechnung_bestellnummern	4	25	25	0,07	0,14	0,14	0,14	29	29
Zero-Shot	frachtbrief_packstueckanzahl	2	1	13	0,13	0,67	0,13	0,22	15	3
Zero-Shot	T1_MRN	3	1	21	0,12	0,75	0,13	0,21	24	4
Zero-Shot	rechnung_einzelpositionen_gesamtanzahl	3	4	21	0,11	0,43	0,13	0,19	24	7
Zero-Shot	lieferschein_einzelpositionen_anzahl	1	5	8	0,07	0,17	0,11	0,13	9	6
Zero-Shot	rechnung_ursprungserklaerung_text	1	1	9	0,09	0,5	0,1	0,17	10	2
Zero-Shot	T1_positionen_insgesamt	2	2	22	0,08	0,5	0,08	0,14	24	4
Zero-Shot	T1_packstücke_insgesamt	1	1	23	0,04	0,5	0,04	0,08	24	2
Zero-Shot	T1_gesamtrohmase	1	2	23	0,04	0,33	0,04	0,07	24	3
Zero-Shot	frachtbrief_versendungsdatum	0	0	14	0	0	0	0	14	0
Zero-Shot	T1_verschluss_feld_19_10	0	0	5	0	0	0	0	5	0
Zero-Shot	T1_wareneingangsnummer_stempel	0	0	0	0	0	0	0	0	0
Zero-Shot	T1_frist	0	0	24	0	0	0	0	24	0

Tabelle B.1.: Klassenspezifische Extraktionsmetriken der Kopfdaten für 30 Sendungen mit der Methode Zero-Shot Prompting (erster Teil).

Prompttyp	Klasse	TP	FP	FN	Accuracy	Precision	Recall	F1	Support	Gpt-4o
Zero-Shot	rechnung_rechnungsdatum	0	0	23	0	0	0	0	23	0
Zero-Shot	rechnung_lieferplan	0	1	0	0	0	0	0	0	1
Zero-Shot	lieferschein_lieferplan	0	1	0	0	0	0	0	0	1
Zero-Shot	lieferschein_ursprungserklaerung_text	0	0	0	0	0	0	0	0	0
Zero-Shot	eori_gb_ukraine_registrierungsnummer	0	4	0	0	0	0	0	0	4
Zero-Shot	atr_feld_4	0	0	1	0	0	0	0	1	0
Zero-Shot	atr_feld_5	0	0	1	0	0	0	0	1	0
Zero-Shot	atr_feld_6	0	0	1	0	0	0	0	1	0
Zero-Shot	atr_feld_12_stempel	0	0	1	0	0	0	0	1	0
Zero-Shot	atr_feld_13_stempel	0	0	1	0	0	0	0	1	0
Zero-Shot	eur1_feld_2	0	0	2	0	0	0	0	2	0
Zero-Shot	eur1_feld_4	0	0	2	0	0	0	0	2	0
Zero-Shot	eur1_feld_5	0	0	2	0	0	0	0	2	0
Zero-Shot	eur1_feld_11_stempel	0	0	2	0	0	0	0	2	0
Zero-Shot	eur1_feld_12_stempel	0	0	2	0	0	0	0	2	0

Tabelle B.2.: Klassenspezifische Extraktionsmetriken der Kopfdaten für 30 Sendungen mit der Methode Zero-Shot Prompting (zweiter Teil).

Prompttyp	Klasse	TP	FP	FN	Accuracy	Precision	Recall	F1	Support	Gpt-4o
Zero-Shot	währung	37	0	5	0,88	1	0,88	0,94	42	37
Zero-Shot	menge	35	4	7	0,76	0,9	0,83	0,86	42	39
Zero-Shot	ursprungsland	9	6	2	0,53	0,6	0,82	0,69	11	15
Zero-Shot	preis	33	5	9	0,7	0,87	0,79	0,83	42	38
Zero-Shot	menge_maßeinheit	27	9	15	0,53	0,75	0,64	0,69	42	36
Zero-Shot	warenbezeichnung	22	15	20	0,39	0,59	0,52	0,56	42	37
Zero-Shot	packstueckart	1	4	1	0,17	0,2	0,5	0,29	2	5
Zero-Shot	artikelnummer	19	17	23	0,32	0,53	0,45	0,49	42	36
Zero-Shot	rohmasse_maßeinheit	13	0	27	0,33	1	0,33	0,49	40	13
Zero-Shot	packstueckanzahl	8	4	31	0,19	0,67	0,21	0,31	39	12
Zero-Shot	rohmasse	9	7	33	0,18	0,56	0,21	0,31	42	16
Zero-Shot	eigenmasse_maßeinheit	1	4	4	0,11	0,2	0,2	0,2	5	5
Zero-Shot	eigenmasse	0	5	5	0	0	0	0	5	5
Zero-Shot	anmelde_und_handelsstatistische_menge	0	0	0	0	0	0	0	0	0
Zero-Shot	beantragte_beguenstigung	0	0	0	0	0	0	0	0	0
Zero-Shot	betriebliche_identifikationsnummer	0	0	0	0	0	0	0	0	0
Zero-Shot	container_nummer	0	0	0	0	0	0	0	0	0
Zero-Shot	packstueckzeichen_und_nummer	0	2	0	0	0	0	0	0	2
Zero-Shot	warennummer	0	15	6	0	0	0	0	6	15

Tabelle B.3.: Klassenspezifische Extraktionsmetriken der Einzelpositionen für 30 Sendungen mit der Methode Zero-Shot Prompting.

C. Anhang C

Im Folgenden sind Screenshots der beiden Systeme auf *Dataiku*⁸ dargestellt. Die Plattform bietet unter anderem das Umsetzen komplexer Anwendungen mit visueller Darstellung.

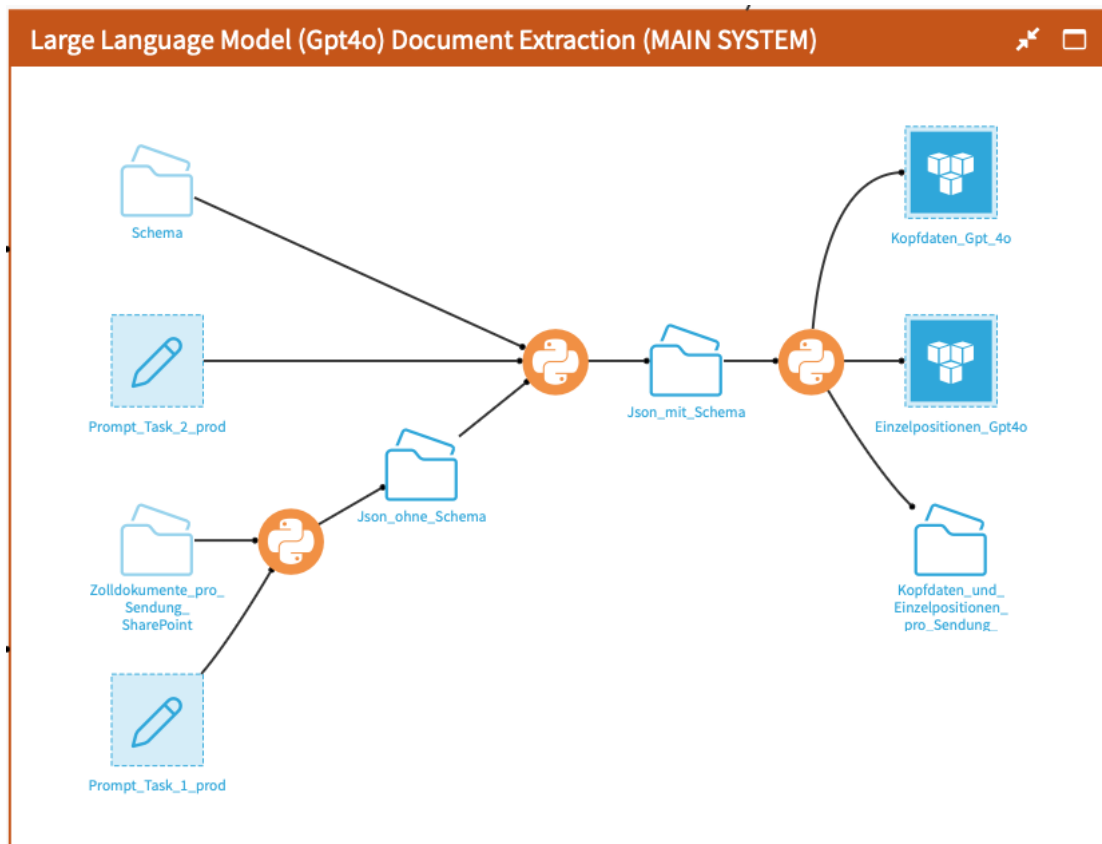


Abbildung C.1.: Screenshot des LLM basierten Systems zur Zolldokumentenextraktion

⁸<https://www.dataiku.com>

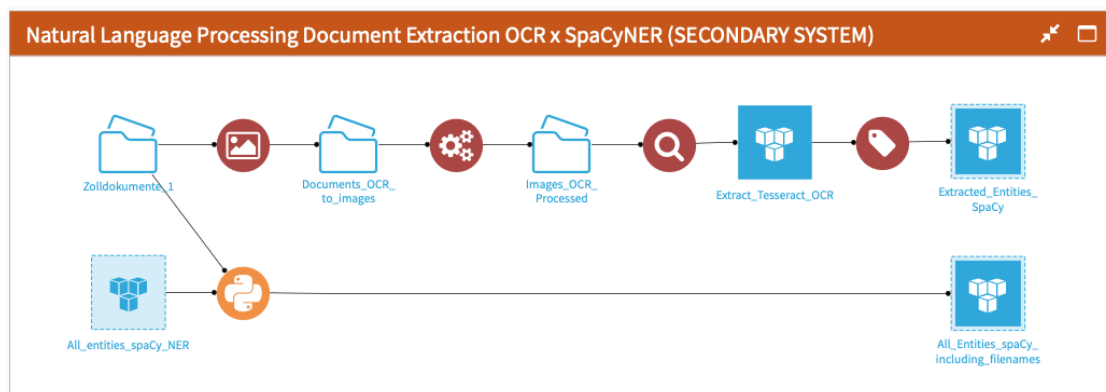


Abbildung C.2.: Screenshot des hybriden klassischen Systems zur Zolldokumentenextraktion