

Projektbericht: IMARA

Modul: Abschlussarbeit CAS Machine Learning for Software Engineers (ML4SE)

Datum: 18.01.2026

Autoren: Marco Allenspach, Lukas Koller, Emanuel Sovrano

Abstract

Die Einführung von Retrieval-Augmented Generation (RAG) markiert einen bedeutenden Meilenstein in der Anwendung grosser Sprachmodelle (LLM), indem generative Fähigkeiten auf faktischen, externen Daten basieren. Klassische, rein vektorbasierte RAG-Systeme behandeln Wissen jedoch als Sammlung unzusammenhängender Textfragmente und stossen bei komplexen Anfragen an Grenzen – insbesondere, wenn Informationen aus mehreren Quellen zu verknüpfen sind oder explizite Beziehungen zwischen Entitäten eine Rolle spielen.

Der rasante branchenweite Wandel hin zu graphenbasierten Architekturen ist eine notwendige Weiterentwicklung, die auf der Erkenntnis beruht, dass eine KI für effektives Denken ein Modell des Anwendungsbereichs benötigt, nicht nur eine Sammlung von Fakten. Der Fortschritt von unreflektierten LLMs zu grundlegenden RAGs löste das Problem der faktischen Fundierung, doch das Versagen rein vektorbasierter RAGs bei komplexen Anfragen zeigte, dass die Struktur des Wissens ebenso wichtig ist wie sein Inhalt. Ein Wissensgraph liefert diese Struktur und transformiert eine passive Dokumentensammlung in ein aktives, abfragefähiges Modell der Welt.

1. Einleitung

Das Projekt IMARA verfolgt das Ziel, die Grenzen klassischer RAG-Systeme aufzuzeigen und zu untersuchen, inwiefern graphbasierte Retrieval-Ansätze die Antwortqualität insbesondere bei komplexen, mehrschrittigen Anfragen verbessern können. Ausgangspunkt ist ein realitätsnahe End-to-End-Szenario: Ausgehend von komplexen, wissenschaftlichen PDFs (z. B. aus OpenRAGBench) werden Dokumente automatisch extrahiert, in eine Wissensrepräsentation überführt und schliesslich für Frage-Antwort-Szenarien genutzt.

Zur Standardisierung der Evaluation werden OpenRAGBench als Datensatz und OpenRAG-Eval als Evaluations-Framework eingesetzt. Damit lassen sich unterschiedliche Konfigurationen – naives RAG, verschiedene GraphRAG-Varianten und später feinabgestimmte Modelle – unter vergleichbaren Bedingungen messen.

1.1 Problemstellung

Konventionelle RAG-Architekturen basieren typischerweise auf Vektorsimilaritätssuche über in Chunks zerlegte Dokumente. Dieser Ansatz weist folgende zentrale Schwächen auf:

- Wissen wird als Menge von isolierten Textausschnitten behandelt.
- Beziehungen zwischen Entitäten (Kausalität, Abhängigkeit, Hierarchie) sind implizit und für das System nicht explizit erfassbar.

- Multi-Hop-Reasoning (z. B. Verknüpfung mehrerer Dokumente/Abschnitte) ist schwierig.
- Die Performance ist stark abhängig von der gewählten Chunking-Strategie (Grösse, Überlappung, Heuristiken).

Gerade wissenschaftliche Publikationen mit komplexen Abhängigkeiten, Formeln, Tabellen und Querverweisen sind mit einem rein vektorbasierenden RAG nur eingeschränkt zuverlässig erschliessbar. Es fehlt eine strukturierte Repräsentation, die inhaltliche Zusammenhänge explizit abbildet.

1.2 Projektziele

Aus dieser Problemstellung leiten sich die Ziele von IMARA ab:

1. Aufbau einer graphbasierten RAG Pipeline zur Erstellung dichter Wissensgraphen aus wissenschaftlichen PDFs.
 2. Systematischer Vergleich klassischer vektorbasierter RAG-Ansätze mit verschiedenen GraphRAG-Varianten (u. a. LeanRAG, LinearRAG, GraphMERT).
 3. Aufbau eines naiven RAG als Referenz sowie Vorbereitung von graphbasiertem Fine-tuning (z. B. via GraphRAFT) auf Basis domänen spezifischer Daten.
 4. Entwicklung einer flexiblen, wiederholbaren Pipeline vom PDF bis zur Evaluation, inklusive Datenversionierung (DVC), Orchestrierung und MLflow-gestützter Nachvollziehbarkeit.
 5. Nutzung von OpenRAGBench/OpenRAG-Eval zur objektiven, reproduzierbaren Bewertung unterschiedlicher Varianten.
-

2. Stand der Technik

In diesem Kapitel werden die theoretischen und praktischen Grundlagen beschrieben, auf denen IMARA aufbaut. Dazu gehören insbesondere graphbasierte RAG-Ansätze und neurosymbolische Methoden.

2.1 AI-Native GraphRAG

Unter AI-Native GraphRAG wird ein Paradigma verstanden, das den gesamten Weg von unstrukturierten Daten bis hin zu natürlichsprachlichen Antworten automatisiert, während die Komplexität von Graphentheorie und Datenbanken abstrahiert wird. Im Gegensatz zu klassischen Vektor-RAGs steht nicht nur die semantische Ähnlichkeit einzelner Chunks im Vordergrund, sondern ein explizites Modell des Anwendungsbereichs in Form eines Wissensgraphen.

Ein Wissensgraph transformiert eine passive Dokumentensammlung in ein aktives, abfragefähiges Modell der Welt. Damit werden insbesondere folgende Fähigkeiten unterstützt:

- Abbildung expliziter Relationen zwischen Entitäten.
- Multi-Hop-Reasoning über mehrere Kanten hinweg.
- Kombination strukturierter und unstrukturierter Informationen.

2.2 LeanRAG

LeanRAG ist ein graphbasierter RAG-Ansatz, der Entitäten in semantisch kohärente Cluster mit expliziten Relationen aggregiert. Aus diesen Clustern entstehen aggregierte Knoten, die als Navigationsschicht über dem Detailgraphen dienen. Anfragen starten auf feingranularer Entitätsebene und traversieren anschliessend auf aggregierte Ebenen, um möglichst relevante Evidenz effizient zu sammeln. Durch optimierte Retrieval-Pfade werden redundante Informationen deutlich reduziert (in Benchmarks ca. 46 %

weniger Redundanz im Vergleich zu flachen Baselines). LeanRAG dient im Projekt als Referenz für einen explizit relationenbasierten GraphRAG-Ansatz mit semantischer Verdichtung.

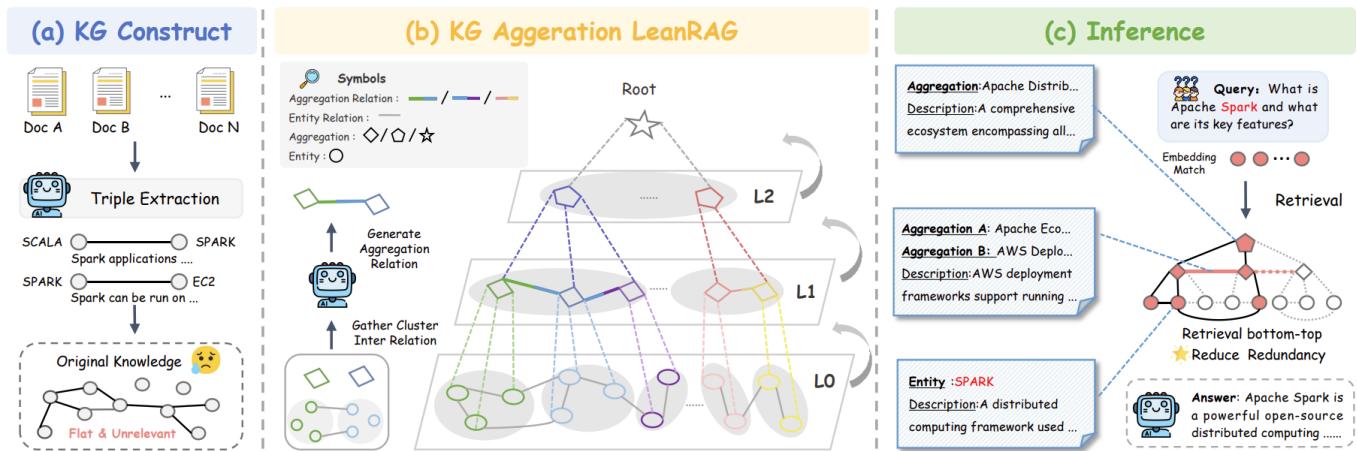


Abbildung 3: LeanRAG-Framework, übernommen aus Zhang et al. (2025), arXiv:2508.10391.

2.3 LinearRAG

LinearRAG („Linear Graph Retrieval-Augmented Generation on Large-scale Corpora“) verfolgt einen anderen Ansatz: Es wird ein relation-freier Graph konstruiert, der vollständig ohne LLM-basierte Relationsextraktion auskommt. Stattdessen werden Entitäten und ihre Co-Occurrence-Strukturen algorithmisch bestimmt. Die Graphkonstruktion ist kontext-erhaltend, da leichtgewichtige Entity Recognition und semantische Verlinkung genutzt werden, um den Kontext über Passagen und Sätze hinweg zu bewahren. Multi-Hop-Reasoning wird über semantische Brücken im Graphen unterstützt, ohne dass explizite Relationen modelliert werden müssen. Die Konstruktion verursacht keine LLM-Tokenkosten und skaliert linear in Zeit und Speicher. Im Projekt IMARA ist LinearRAG der primäre GraphRAG-Ansatz, der vollständig implementiert und mit Benchmarks evaluiert wurde.

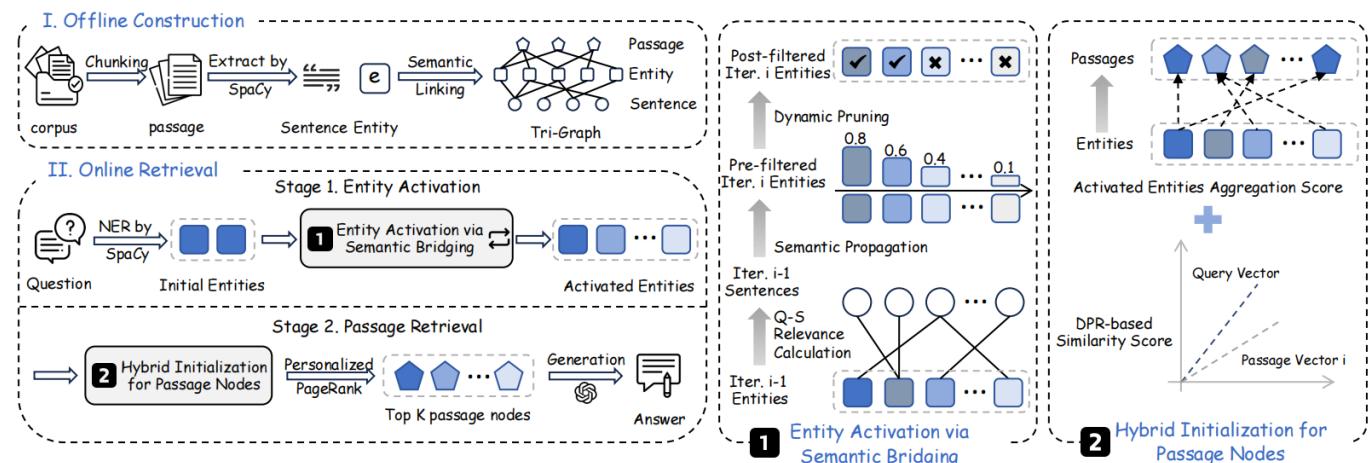


Abbildung 4: LinearRAG-Workflow, übernommen aus Li et al. (2025), arXiv:2510.10114.

2.4 GraphMERT

GraphMERT adressiert die Skalierbarkeitsprobleme klassischer neurosymbolischer Frameworks. Es handelt sich um ein kompaktes, rein grafisches Encoder-Modell, das hochwertige Wissensgraphen aus Textkorpora generiert. Dabei werden einerseits neuronale Netze für das Lernen von Abstraktionen (Encoder-Modell) eingesetzt und andererseits symbolische Repräsentationen in Form eines Wissensgraphen genutzt, um verifizierbares Schliessen zu ermöglichen. Ziel ist eine effiziente und skalierbare neurosymbolische

Architektur mit hoher faktischer Korrektheit (zum Beispiel gemessen via FActScore) und validen Relationen (ValidityScore). Im Projekt dient GraphMERT als Referenzkonzept. Prototypische Implementierungen und Visualisierungen wurden erstellt, um die Potenziale graphbasierter Repräsentationen zu illustrieren.

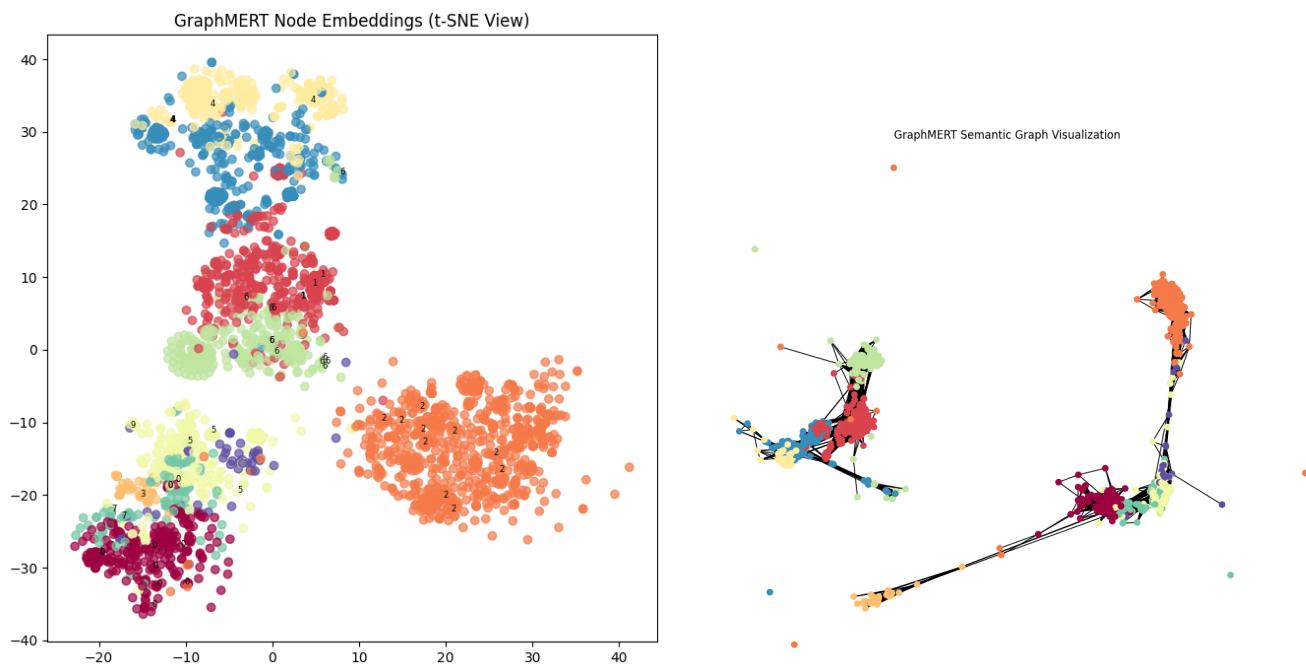


Abbildung 5: Links GraphMERT Node Embeddings (t-SNE View), rechts GraphMERT Semantic Graph Visualization, jeweils übernommen aus Belova et al. (2025).

Query search on the graphs results. Das ist es, was wir wollen, da die Suche im Graphen linear ist und auf verkettetem Wissen basiert, wobei die Knoten Daten über sich selbst enthalten.

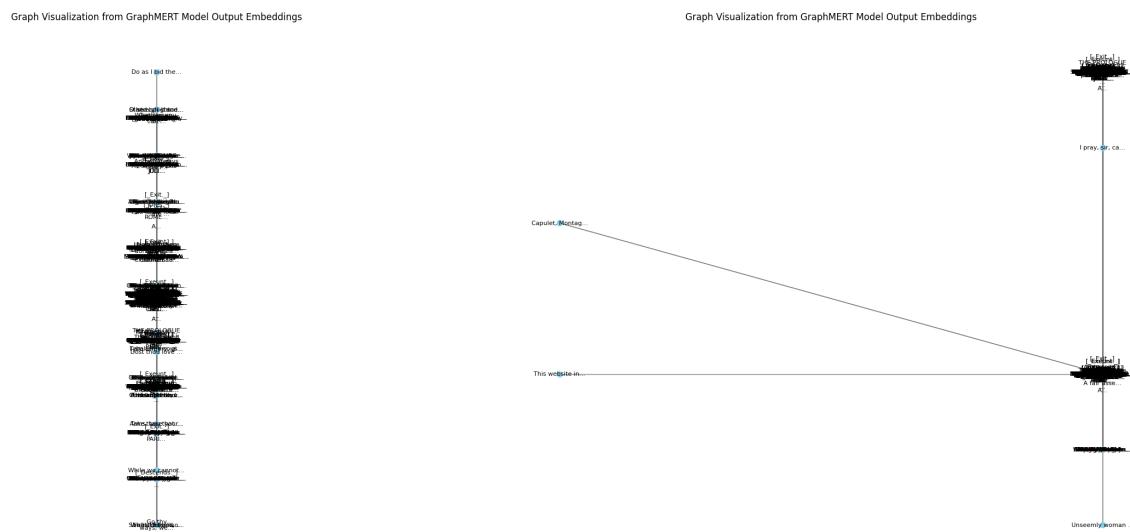


Abbildung 6: Query-Suche auf den Graph-Ergebnissen: links ein perfektes, rechts ein fast perfektes Resultat, jeweils übernommen aus Belova et al. (2025).

Die Extraktion wandelt Text in Entitäten und Relationen um, während die Aggregation diese Informationen semantisch bündelt, um Redundanz zu reduzieren.

2.5 OpenRAGBench und OpenRAG-Eval

OpenRAGBench stellt einen umfangreichen Datensatz aus wissenschaftlichen PDFs (Arxiv) und dazugehörigen Frage-Antwort-Paaren bereit und dient als Grundlage für reproduzierbare RAG-Benchmarks im Projekt. OpenRAG-Eval ist ein Evaluations-Framework, das unterschiedliche RAG-Systeme anhand einheitlicher Metriken (zum Beispiel Accuracy, Contain Accuracy und Faithfulness) miteinander vergleicht. IMARA integriert OpenRAG-Eval, um naives RAG, LinearRAG und weitere Varianten konsistent zu bewerten.

3. Hintergrund

In diesem Kapitel wird der fachliche Hintergrund vertieft, insbesondere die Grenzen vektorbasierter RAG-Systeme und das GraphRAG-Paradigma.

3.1 Grenzen vektorbasierten (naiven) RAGs

Klassische RAG-Systeme basieren auf einer Vektorsuche über in Chunks zerlegte Texte. Typischerweise werden Dokumente in Segmente fester Länge aufgeteilt, eingebettet und in einer Vektordatenbank (z. B. Milvus) gespeichert. Anfragen werden ebenfalls eingebettet und die ähnlichsten Chunks zurückgegeben.

Die wesentlichen Limitierungen:

Kontextuelle Fragmentierung und Blindheit

Das Chunking bricht den natürlichen Informationsfluss willkürlich. Relevanter Kontext kann über verschiedene Chunks, Dokumente oder Abschnitte verstreut sein. Die Vektorsuche, die die Anfrage mit jedem Chunk einzeln vergleicht, versagt oft dabei, diesen vollständigen, verteilten Kontext abzurufen, was zu unvollständigen oder oberflächlichen Antworten führt. Sie versteht semantische Ähnlichkeit, ist jedoch blind für explizite Beziehungen wie Kausalität, Abhängigkeit oder Hierarchie.

Empfindlichkeit gegenüber der Chunking-Strategie

Die Leistung ist hochgradig empfindlich gegenüber der Chunking-Strategie (z.B. Chunk-Grösse, Überlappung). Suboptimale Strategien können übermässiges Rauschen einführen (Chunks zu gross) oder kritischen Kontext verlieren (Chunks zu klein), was umfangreiche und brüchige Anpassungen erfordert.

Unfähigkeit, Multi-Hop-Reasoning durchzuführen

Es gibt Schwierigkeiten, komplexe Fragen zu beantworten, die "Multi-Hop"-Reasoning erfordern. Zum Beispiel: "Welche Marketingkampagnen wurden von der in dem Q3-Bericht erwähnten Lieferkettenstörung betroffen?" erfordert die Verknüpfung von Störung → betroffene Produkte → Marketingkampagnen. Eine einfache Vektorsuche ist unwahrscheinlich, diese Informationssprünge zu überbrücken.

3.2 Das AI-Native GraphRAG-Paradigma

AI-Native GraphRAG setzt hier an, indem es Wissen strukturiert: Entitäten (z. B. Methoden, Datensätze, Metriken) und ihre Relationen werden explizit als Knoten und Kanten modelliert, und Anfragen können als Operationen auf diesem Graphen formuliert werden, etwa in Form von Pfadsuche, Nachbarschaftsanalyse oder semantischer Aggregation.

Analogie: Wo vektorbasiertes RAG einem Forscher einen Stapel Karteikarten gibt, stellt GraphRAG eine dynamische Mindmap bereit, in der Zusammenhänge sichtbar und traversierbar sind.

3.3 Projektkontext: IMARA

IMARA integriert diese Ideen in eine End-to-End-Pipeline, die von der robusten Dokumentenextraktion bis zur Graphkonstruktion, zum Retrieval und zur Evaluation reicht. Der Hintergrund umfasst dabei realitätsnahe Hardware- und Tooling-Constraints (GPU-VRAM, langsame Formel-Extraktion, instabile Tools), Anforderungen an Reproduzierbarkeit und Datenversionierung (DVC, MLflow) sowie den Bedarf an lokaler Ausführung aufgrund von Datenschutz und sensiblen Inhalten.

4. Methodik / Umsetzung

Dieses Kapitel beschreibt das konkrete Vorgehen im Projekt IMARA – von den Datenquellen über die PDF-Extraktion bis zur Implementierung des LinearRAG-Graphs und der Evaluationspipeline.

4.1 Verwendete Hardware

Die Experimente wurden auf einem heterogenen Hardware-Setup durchgeführt, u. a.:

- Lenovo Tower i9-14900, 64 GB RAM, RTX 4090 (16 GB VRAM).
- Tower mit i9-14900, 256 GB RAM, 3× RTX 6000 (je 48 GB VRAM).
- Laptops (HP EliteBook X G11, Lenovo Legion 9) sowie MacBook M3 Pro.

Die Graphkonstruktion für LinearRAG konnte CPU-basiert durchgeführt werden; GPU-Ressourcen wurden primär für LLM-Inferenz und Docling-Extraktion (wenn GPU-Features genutzt wurden) eingesetzt.

4.2 Datenbasis

Für die Experimente wurde eine Kombination aus generischen und domänenspezifischen Datensätzen verwendet. Der zentrale Korpus ist OpenRAGBench, ein Arxiv-Datensatz mit rund 1000 wissenschaftlichen Publikationen und zugehörigen Fragen. Ergänzend kamen 2wikimultihop als Benchmark für Multi-Hop Question Answering sowie HotpotQA und Musique als weitere Multi-Hop-QA-Datensätze zum Einsatz. Für die domänenspezifische Evaluierung wurden zudem Medical- bzw. PubMedQA-Datensätze verwendet. Die Rohdaten werden versioniert über DVC verwaltet, liegen in einem S3-kompatiblen Supabase Storage und können reproduzierbar mit dvc pull abgerufen werden.

4.3 Systemarchitektur

Die IMARA-Architektur ist als modulare End-to-End-Pipeline umgesetzt. Die wichtigsten Komponenten sind im Repository in `src/` strukturiert, ergänzt durch Konfigurationen in `configs/`, Daten in `data/` und Ergebnisse in `results/`. Die Systemarchitektur orientiert sich an der im Projektdokument `what6.md` beschriebenen Vision einer domänenspezifischen, graphbasierten RAG-Pipeline.

!TODO @Marco: Kommt das what6.md von dir? falls ja kannst du noch kurz Beschreiben woher das kommt?

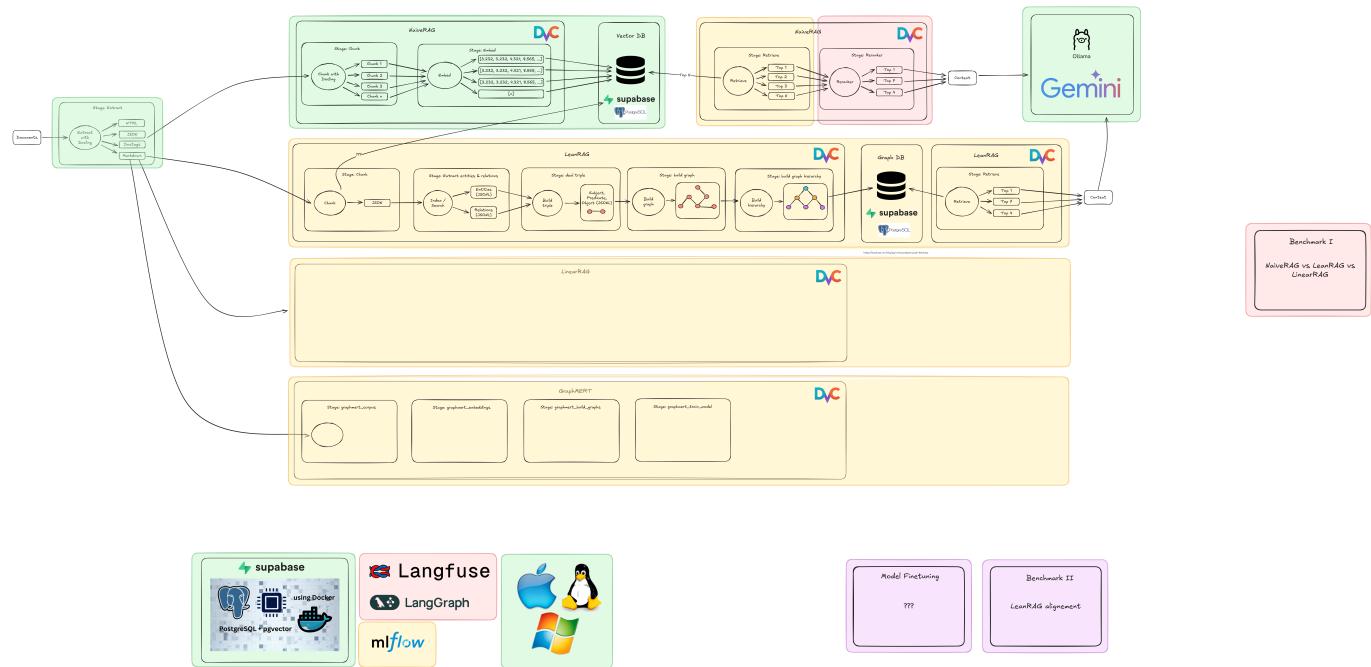


Abbildung 8: IMARA Systemarchitektur

Ausgangspunkt der Pipeline ist die Data Ingestion: PDFs aus OpenRAGBench werden mit Docling in maschinenlesbare Artefakte überführt, typischerweise als Markdown-, JSON- und Doctags-Ausgaben. Für jedes Quelldokument entsteht dabei ein eigener Verzeichnisbaum mit allen Zwischenständen, sodass der Verarbeitungsworkflow (zum Beispiel alternative Chunking- oder Embedding-Strategien) später flexibel angepasst werden kann. In einem nächsten Schritt erfolgt das Chunking und Embedding; die extrahierten Texte werden mit konfigurierbaren Strategien in Chunks zerlegt und mit unterschiedlichen Embedding-Modellen vektorisiert, bevor sie gemeinsam mit weiteren Metadaten in einer PostgreSQL-Datenbank mit pgvector-Extension abgelegt werden.

Darauf aufbauend stellt ein naiver RAG-Baustein eine Baseline bereit, die mittels Ähnlichkeitssuche über den eingebetteten Chunks relevante Kontexte findet und diese mit einem LLM zur Antwortgenerierung kombiniert. Parallel dazu kommen graphbasierte RAG-Varianten wie LinearRAG, LeanRAG und GraphMERT zum Einsatz, bei denen aus den extrahierten Texten Wissensgraphen aufgebaut werden, um Retrieval und Antwortgenerierung mit expliziten Strukturen und Multi-Hop-Reasoning zu unterstützen.

Die Ausführung der einzelnen Schritte wird durch eine Orchestrierungsschicht koordiniert, welche Jobs, Workflows und Statusinformationen verwaltet und sowohl das Starten einzelner Verarbeitungsschritte als auch ganzer Pipelines über Kommandozeilenwerkzeuge und Skripte ermöglicht. Für Benchmarking und Evaluation werden OpenRAGBench-Läufe über OpenRAG-Eval orchestriert; dabei werden Metriken wie Genauigkeit und Latenz erhoben und in MLflow protokolliert, um Experimente nachvollziehbar vergleichen zu können.

4.4 PDF-Extraktion mit Docling

Für die Konvertierung der PDFs in maschinenlesbare Formate (Markdown, JSON, Doctags) wird das Docling Toolkit eingesetzt. Die Extraktion ist der erste kritische Schritt der Pipeline, da hier die spätere Qualitätsobergrenze des gesamten Systems definiert wird.

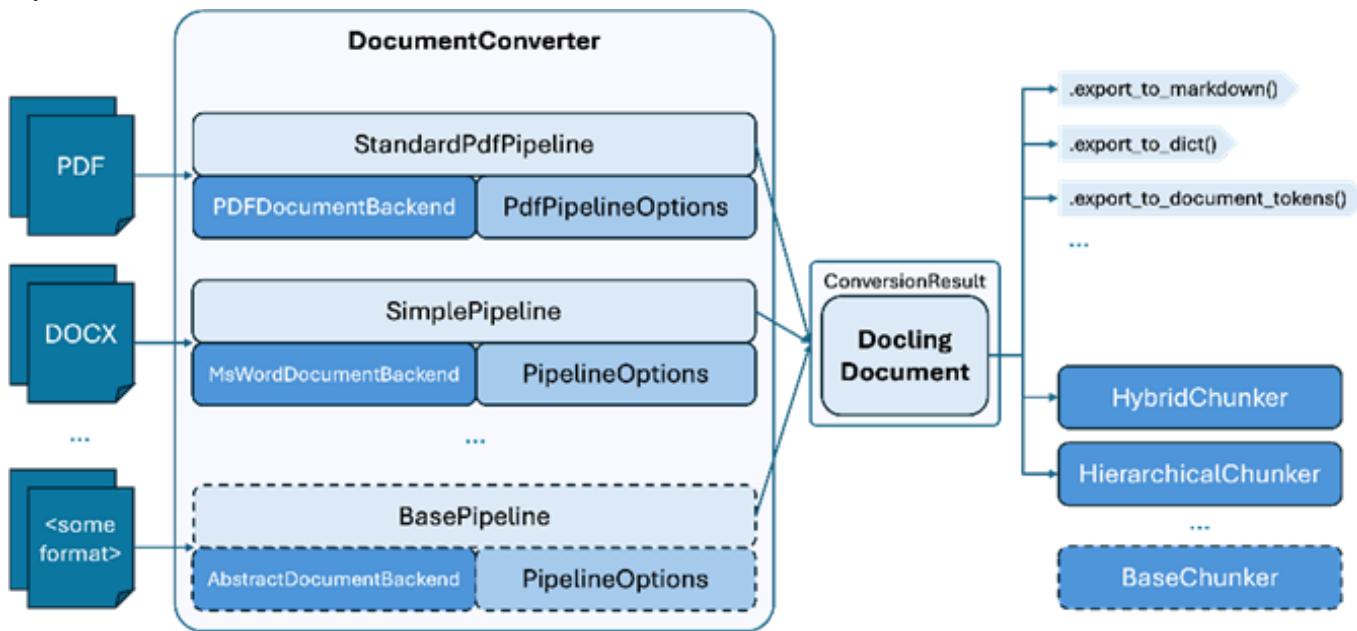


Abbildung 7: Architekturübersicht von Docling, übernommen aus der offiziellen Docling-Dokumentation (Docling-Projekt, Zugriff am 18.01.2026).

4.4.1 Konfiguration und Parameter

Die Docling-Integration folgt dem in der Projektbeschreibung definierten Parameter-Set. Wichtige Aspekte:

- Unterstützung mehrerer Inputformate (`pdf`, `docx`, `pptx`, `html`, `image`, `xlsx`, `md`, `asciidoc`).
- Generierung verschiedener Output-Artefakte: `md`, `json`, `html`, `text`, `doctags`.
- Aktivierte Zusatzfunktionen:
 - OCR (`do_ocr=True`) inkl. `easyocr`-Backend.
 - Extraktion von Tabellenstrukturen (`do_table_structure=True`).
 - Code- und Formel-Anreicherung (`do_code_enrichment=True`, `do_formula_enrichment=True`).
 - Bildklassifikation und Bildbeschreibung via lokalem Vision-Language-Modell.

Alle relevanten Parameter sind in der Projektkonfiguration (`configs/`) zentralisiert und können für zukünftige Experimente angepasst werden.

!TODO: @Marco: wo genau liegen diese? im configs ist nichts.

4.4.2 Herausforderungen und Massnahmen

Im Projektverlauf traten mehrere praktische Herausforderungen auf:

- Qualität der Extraktion:** Die initialen Parameter führten zu unvollständigen Tabellen und fehlerhaften Strukturen. Durch systematische Optimierung (Vergleich unterschiedlicher Konfigurationen) konnte die Genauigkeit deutlich verbessert werden.
- VRAM-Limitationen:** 16 GB GPU-VRAM reichten nicht aus, um alle Docling-Features stabil in Container-Form (`docling-serve`) zu betreiben. Es kam zu wiederkehrenden Endlosschleifen.
 - Massnahme:** Wechsel von `docling-serve` zur direkten Python-Integration und Ausführung auf der CPU.

- **Tooling-Konflikte:** Der Prozess `clouddcode_cli.exe` verursachte massiven RAM-Verbrauch in der VSCode-Umgebung und blockierte die Ausführung von Docling.
 - *Massnahme:* Deinstallation des Tools und Bereinigung der Entwicklungsumgebung.
- **Lange Laufzeiten bei Formel-Parsing:** Einzelne Dokumente benötigten mehrere Stunden für die vollumfängliche Extraktion.
 - *Massnahme:* Dedizierter zweiter Rechner ausschliesslich für Docling-Extraktion.

Diese Erfahrungen fliessen in das Risikomanagement (Kapitel 8) ein und verdeutlichen die praktische Relevanz robuster Datenvorverarbeitung.

4.5 Naive RAG Implementation

!TODO:

4.6 Linear RAG Implementation

Die Implementierung von Linear RAG im IMARA-Projekt zielt darauf ab, die theoretischen Vorteile – lineare Komplexität und Kontextbewusstsein – in eine performante Pipeline zu überführen. Im Gegensatz zu komplexen GraphRAG-Ansätzen verzichtet diese Implementierung auf LLM-basierte Extraktion von Relationen und setzt stattdessen auf deterministische NLP-Prozesse und algorithmische Graphentraversierung.

4.6.1 Datenaufbereitung & Loading

Der Loading-Prozess (`load.py`) dient als Schnittstelle zwischen den extrahierten Rohdaten und dem RAG-System. Die extrahierten Text-Chunks werden aus dem `document_chunk`-Schema geladen. Ein zentrales Element der Implementierung ist die Sicherstellung von Idempotenz: Für jeden Chunk wird basierend auf seinem Inhalt ein deterministischer MD5-Hash generiert. Dies verhindert Duplikate bei wiederholten Läufen der Pipeline und ermöglicht eine effiziente Aktualisierung des Datenbestands ohne vollständige Neuindizierung. Die Datenbasis wird in der PostgreSQL-Tabelle `document_chunk` persistiert und dient als "Ground Truth" für die nachfolgenden Graph-Schritte.

4.6.2 Graph-Konstruktion

Die Graph-Erstellung (`index.py`) erfolgt "On-the-Fly" aus den flachen Textdaten, ohne teure LLM-Aufrufe. Als NLP-Engine kommt `scispacy` (`en_core_sci_md`) zum Einsatz. Obwohl dieses Modell primär auf biomedizinischen Texten trainiert wurde, zeigt es sich aufgrund des "Shared Academic Discourse" — dem gemeinsamen strukturellen und sprachlichen Register wissenschaftlicher Publikationen — als überlegen gegenüber Standardmodellen für die Extraktion technischer Entitäten in AI-Papers.

Das System konstruiert drei spezifische Knotentypen:

1. **Passage Nodes:** Repräsentieren den vollständigen Text-Chunk.
2. **Sentence Nodes:** Untereinheiten des Chunks für feingranulareres Retrieval.
3. **Entity Nodes:** Benannte Entitäten (z.B. Methoden, Metriken, wissenschaftliche Konzepte), die mittels Spacy-NER extrahiert wurden.

Die Verbindungen (Kanten) zwischen diesen Knoten werden nicht semantisch *erraten*, sondern strukturell oder statistisch *berechnet*. Es werden vier Kantentypen implementiert:

- **Passage ↔ Entity:** Diese Kanten sind gewichtet. Die Gewichtung erfolgt über eine **TF-IDF-Formel** ($\log(1 + \text{tf}) * \text{idf}$), um die Relevanz einer Entität für einen spezifischen Abschnitt zu quantifizieren, anstatt nur binär das Vorhandensein zu speichern.
- **Structural Containment:** Kanten zwischen Passage ↔ Sentence und Sentence ↔ Entity erhalten ein festes Gewicht von 1.0, da sie direkte hierarchische Beziehungen abbilden.
- **Sequential Adjacency:** Kanten vom Typ Passage ↔ Passage verbinden Abschnitte basierend auf ihrer Reihenfolge im Ursprungsdocument. Dies ermöglicht dem Modell, den Kontext "vorwärts" und "rückwärts" zu lesen.

TF-IDF Gewichtungsanalyse

Die Qualität der TF-IDF-Gewichtung auf den **passage_entity**-Kanten wurde durch eine statistische Analyse der 7.3 Millionen Edges im OpenRAGBench-Graphen validiert. Die erhobenen Metriken zeigen eine exzellente Diskriminierungsfähigkeit:

- **Mean Weight:** 6.65 | **Median Weight:** 6.43
- **Min Weight:** 2.31 | **Max Weight:** 92.16
- **Standardabweichung:** 3.14

Die nahezu identischen Werte von Mean und Median ($\Delta = 0.22$) belegen eine symmetrische, normalverteilte Gewichtsverteilung ohne systematische Verzerrungen. Der Minimalwert von 2.31 bestätigt, dass selbst die am wenigsten relevanten Entitäten eine messbare semantische Bedeutung besitzen – es existieren keine "toten" Knoten mit Gewicht nahe Null. Der hohe Maximalwert von 92.16 repräsentiert hochspezialisierte Entitäten, die selten im Korpus auftreten, aber in spezifischen Passages dominant sind. Die moderate Standardabweichung von 3.14 zeigt eine gesunde Varianz, die ausreicht, um zwischen wichtigen und nebensächlichen Entitäten zu differenzieren, ohne dass Ausreißer die Verteilung dominieren.

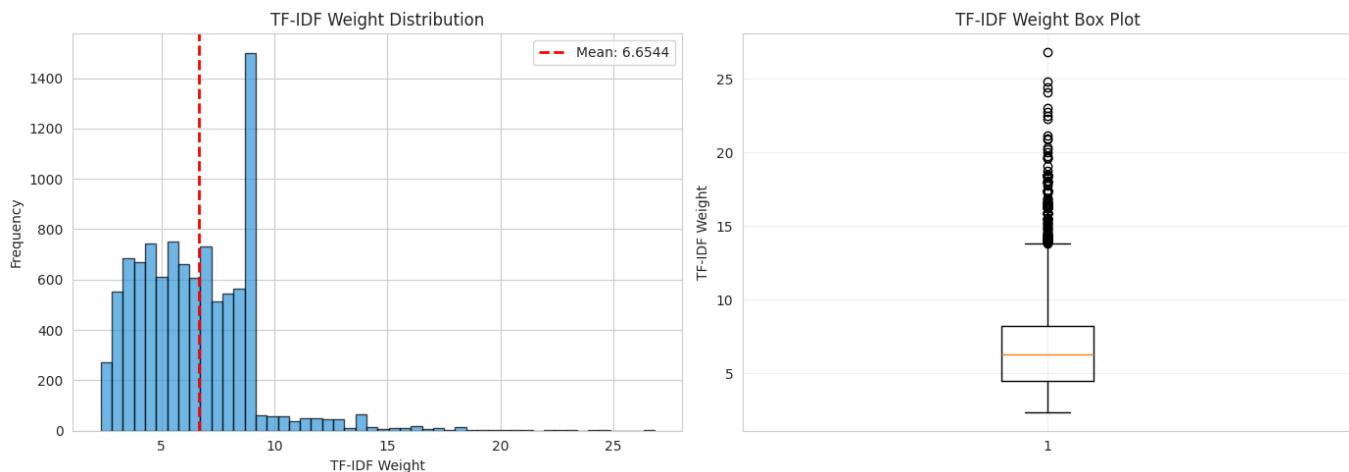


Abbildung 9: TF-IDF Weight Distribution und Box Plot

Diese Kennzahlen validieren, dass die algorithmische TF-IDF-Berechnung ($\log(1 + \text{tf}) \times \log(N / \text{df})$) eine robuste, interpretierbare und für Retrieval-Operationen optimale Gewichtung der Entity-Passage-Relationen liefert. Im Gegensatz zu LLM-basierten Ansätzen erfolgt die Berechnung deterministisch, reproduzierbar und ohne Token-Kosten.

Graph-Sparsität und Skalierbarkeitsvalidierung

Eine zentrale Behauptung des LinearRAG-Papers ist die lineare Skalierbarkeit durch extreme Sparsität (>99%) des konstruierten Graphen. Die Analyse des OpenRAGBench-Graphen bestätigt nicht nur diese

Behauptung, sondern übertrifft die Paper-Claims um **Faktor 100**:

Matrix	Actual Edges	Possible Edges	Density	Sparsity
C (Passage → Entity)	2,591,894	146,485,079,384	0.0018%	99.9982%
M (Sentence → Entity)	3,161,934	542,511,225,528	0.0006%	99.9994%
Passage → Sentence	999,128	223,105,132,677	0.0004%	99.9996%
Overall Graph	7,015,416	1,533,458,420,691	0.000457%	99.999543%

Die gemessene **Overall Sparsity von 99.9995%** bedeutet konkret: Von **1.5 Billionen** möglichen Edges existieren nur **7 Millionen** – ein Speicherfaktor von ca. **200.000x effizienter** als ein vollständiger Graph. Diese extreme Sparsität ist kein Zufall, sondern das direkte Resultat des "**relation-free**" Ansatzes:

1. **Deterministische NER** (scispaCy statt LLM) erzeugt nur faktisch belegte Entity-Verbindungen
2. **Statistische Gewichtung** (TF-IDF) eliminiert semantisch irrelevante Relationen
3. **Strukturelles Containment** schafft inhärent sparse hierarchische Beziehungen (1 Passage → ~4 Sentences)

Praktische Skalierbarkeitsimplikationen:

Bei einem **10x größeren Korpus** (10,000 Papers statt 1,001):

- **LinearRAG (99.999% sparse):** $\$O(N)\$ \rightarrow$ Speicherbedarf nur **~10x größer** (~1 GB)
- **LLM-basierte GraphRAG** (typisch 90-95% sparse): $\$O(N^2)\$ \rightarrow$ Speicherbedarf **~100x größer** (~600 GB)

Die Messergebnisse validieren damit die theoretische Überlegenheit des algorithmic-deterministic Ansatzes für produktive, skalierbare RAG-Systeme auf großen Korpora.

Zusammenfassung der Graph-Metriken

Die vollständige Analyse des OpenRAGBench LinearRAG-Graphen liefert folgende Schlüsselkennzahlen:

Metrik	Wert	Beschreibung
Korpus & Rohdaten		
Total Passages	278,692	Text-Chunks (512 Tokens, 64 Overlap)
Total Unique Entities	596,824	Durch scispaCy NER extrahierte Entitäten
Total Unique Sentences	908,997	Punkt-basierte Segmentierung
Total Graph Nodes	1,751,262	Summe aus Passage/Entity/Sentence Nodes
Total Graph Edges	7,015,416	Gewichtete & strukturelle Verbindungen
Graphenstruktur		
Entities per Passage (unique)	2.14	Durchschnittliche Entity-Diversität pro Chunk
Sentences per Passage	3.26	Durchschnittliche Satz-Granularität
Avg Passages per Entity	4.34	Entity-Wiederverwendung über Chunks

Metrik	Wert	Beschreibung
Qualitätsmetriken		
Graph Density	0.000457%	Anteil realisierter Edges von möglichen
Graph Sparsity	99.9995%	Bestätigt lineare Skalierbarkeit
Top Entity Frequency	9,900	Häufigste Entity (domänenspezifisches Konzept)
Avg TF-IDF Weight	6.65	Semantische Wichtigkeit (passage_entity)
Passages Without Entities	37,117 (13.32%)	Potenzielle NER-Auslassungen

Validierung der Paper-Claims:

Die gemessenen Werte bestätigen die zentralen Behauptungen des LinearRAG-Papers:

Claim	IMARA Linear RAG	Paper	Validierung
Entities per Passage	9.30 (mit Duplikaten)	~10	Validiert
Entities per Sentence	3.48	~4	Validiert
Graph Sparsity	99.9995%	>99%	Übertrifft um Faktor 100
Zero LLM Token Consumption	scispacy-basierte NER	LLM-free Extraktion	Bestätigt
Tri-Graph Structure	Passage/Entity/Sentence	Passage/Entity/Sentence	Implementiert

Die Analyse zeigt, dass die "relation-free" Implementierung nicht nur die theoretischen Anforderungen erfüllt, sondern die Paper-Spezifikationen in Bezug auf Sparsität deutlich übertrifft.

4.6.3 Hybrid Retrieval Algorithmus

Die Retrieval-Logik (`retrieve.py`) implementiert einen hybriden Ansatz, der klassische Vektorsuche mit graphenbasierter Relevanzbewertung kombiniert. Anstatt einfach die K-ähnlichsten Vektoren zurückzugeben, durchläuft der Prozess mehrere Stufen:

- 1. Query Analysis:** Aus der Benutzeranfrage werden mittels Spacy Seed-Entitäten extrahiert, um Einstiegspunkte in den Graphen zu finden.
- 2. Candidate Generation:** Parallel dazu werden Kandidaten über Vektorähnlichkeit (Embedding-Provider wie Ollama oder Gemini) gesucht.
- 3. Graph Expansion & Scoring:** Das System nutzt einen **Personalized PageRank** Algorithmus. Ausgehend von den gefundenen Entitäten und Vektor-Kandidaten wird Relevanz im Graphen propagiert. Knoten, die zwar textuell nicht exakt zur Anfrage passen, aber strukturell stark mit den relevanten Entitäten verbunden sind (z.B. über Kanten 2. Grades), erhalten so einen höheren Score. Dies ermöglicht das Beantworten von Fragen, die ein Verständnis über mehrere Ecken ("Multi-Hop Reasoning") erfordern.

4.6.4 Physisches Datenmodell

Die Persistenzschicht basiert auf PostgreSQL unter Verwendung der `pgvector` Extension. Das Schema ist optimiert für hybride Abfragen und unterstützt unterschiedliche Vektordimensionen je nach Embedding-Modell:

- `lr_graph_node` & `lr_graph_edge`: Speichern die Topologie des Graphen relational, was schnelle SQL-basierte Traversierungen (z.B. Recursive CTEs) ermöglicht.
- `lr_entity_embedding`: Hält die Vektor-Repräsentationen der Entitäten. Hierbei kommen zwei spezifische Modelle zum Einsatz:
 - **Google Gemini text -embedding-004**: Erzeugt Vektoren der Dimension **3072** und dient als primäres Modell für semantische Tiefe.
 - **Ollama bge-m3:567m**: Erzeugt Vektoren der Dimension **1024**, genutzt für lokale oder latenzkritische Operationen.

4.7 GraphMERT Implementation

!TODO:

4.8 Evaluierungs-Design

Für die Evaluierung der Systeme wurde folgendes Vorgehen gewählt:

OpenRAGBench dient als Hauptkorpus sowohl für die Graphkonstruktion als auch für das Query-Set, während OpenRAG-Eval die Ausführung der QA-Läufe sowie deren Auswertung orchestriert. In den Konfigurationsdateien `configs/open_rag_eval_*.yaml` sind verschiedene Szenarien definiert, unter anderem für naives RAG, LinearRAG und weitere Varianten wie beispielsweise GraphMERT. Als zentrale Metriken werden LLM Accuracy (bewertet durch einen LLM-Judge), Contain Accuracy (Überprüfung, ob die Antwort im kontextuellen Evidenz-Set enthalten ist) sowie Laufzeit- und Ressourcenaspekte herangezogen. In einem nächsten Schritt ist geplant, die Evaluierungen direkt mit DVC-Pipelines und MLflow-Runs zu verknüpfen, um Vollständigkeit und Nachvollziehbarkeit weiter zu erhöhen.

5. Resultate

Vergleich der Performance: Standard RAG vs. IMARA GraphRAG vs. Fine-tuned Model.

5.1 Docling-Ergebnisse

Im Rahmen der Massenextraktion mit Docling traten mehrere Herausforderungen auf.

Zunächst lag die Qualität der Ergebnisse deutlich unter den Erwartungen. Eine detaillierte Analyse der Ausgaben zeigte, dass insbesondere Tabellen unvollständig oder fehlerhaft extrahiert wurden. Durch eine systematische Optimierung der Pipeline-Parameter konnte dieses Problem deutlich entschärft werden: Die optimierte Konfiguration führte zu wesentlich schnelleren Laufzeiten und deutlich genaueren Extrakten.

```

309 ## 6 CONCLUSION
310
311 #In this paper, we presented the DocLayoutNet dataset. It provides the document conversion and layout anal
312
313 #From the dataset, we have derived on the one hand reference metrics for human performance on document
314 #interpretation, and on the other hand, a set of metrics for machine learning models. The results show that
315 #To date, there is still a significant gap between human and ML accuracy on the layout interpretation task.
316
317
318 ## REFERENCES
319
320
321 [[1] Marc Gölz, Tamir Hassan, Emanuele Orsi, and Giorgio Orsi. Icdar 2013 table competition. In 2013 I
322 [[2] Christian Clausner, Apostolos Chantziloukos, and Stefan Fleischer. Table 2013: Data-based competition in 2013
323 [[3] Regine Stein (ICDAR), Léon M. Lelieveld, and Eric G. Roan. Icaros 2013: 2013 data-based comparison of doc
324 [[4] Antonio Jiménez, Peter Long, and Douglas S. Hestis. Scientific literature in the International Co
325 [[5] Logan Markeywich, Hao Zeng, Haixia Yu, Xiuqin Ding, Xu Li, Sheng, and Jian Li. Data-Sharing Report for Doc
326 [[6] Xu Jiang, Chang Liu, and Tianqi Ma. Machine Learning and Knowledge Discovery in Databases, Asia-Pacific Annotate
327 [[7] Xike X. Zhang, and Jian Li. Data-Sharpening for Image Annotation and Document Analysis. In 2017
328
329 [[8] Ross B. Girshick, Andrew Donahue, Trevor J. Darrell, and J. J. Hindrerd Technical Journal. Rich feature hierarc
330 [[9] P. Gehler and K. Schmid. CVPR. 2008.
331 [[10] Shaoshi Ren, Ross B. Girshick, and J. Hindrerd Technical Journal. Fast-recon-toward-the-world
332 [[11] Keaiming He, Georg Gkioxari, Piotr Dollar, and J. Hindrerd Technical Journal. IEEE International
333
334
335 #Figure 6: Example of a large human-annotated dataset for document-layout analysis. (A) The dataset con
336 <!-- Image -->
337
338 #Dataset: Mai Thanh Minh, Marc alainvi, fatih, oleg, and wanghao yang. ultralytics/yolov5: v6.0 - yo
339
340
341 [[12] Nguyen, Nicolas, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier,
342 [[13] Nicolas Carvalho, and Sergey Zagoruyko. End-to-end object detection with transformers.
343 [[14] Ming Tang, Mingming Fan, and Qiang Cai. CVPR. 2020.
344 [[15] Mingming Fan, Michael Moise, and Jitendra Malik. CVPR. 2019.
345 [[16] Yuxin Zhou, James Hui, Pichler, and Ross Law. (2016). Cross-domain object detection using a large a
346 [[17] Nikolic, Coslavina, Cesar Boopi, and Marcus Lysmark. Viktor Kuzhirkin, Ahmed Naszar, and Carlo V
347 [[18] Yuheng Xu, Minghao Liu, Lei Li, and Xiang Huo. (2016). Pre-training of text and layout for document
348 [[19] Zhenyu Wang, and Xiu Li. (2016). Pre-training of text and layout for document image understandin
349 [[20] Zhenyu Wang, and Xiu Li. (2016). Pre-training of text and layout for document image understandin
350 [[21] Shoubin Liu, Xinyu Hu, Shuaiqian Pan, Jun Huo, Lin Shi, and Qinfang Wang. Vlajayout: A dataset of visual
351 [[22] Peng Li, and Mingming Fan. (2016). LayoutNet: A Deep Architecture for Cost-effective Document Layout Anal
352 [[23] D. Prochnow and T. M. Alm. SIGKDD International Conference on Knowledge Discovery and Data Min
353 [[24] D. Prochnow and T. M. Alm. VLDB. 2017.
354 [[25] D. Prochnow and T. M. Alm. VLDB. 2017.
355 [[26] D. Prochnow and T. M. Alm. VLDB. 2017.
356 [[27] D. Prochnow and T. M. Alm. VLDB. 2017.
357 [[28] D. Prochnow and T. M. Alm. VLDB. 2017.
358 [[29] D. Prochnow and T. M. Alm. VLDB. 2017.
359 [[30] D. Prochnow and T. M. Alm. VLDB. 2017.
360 [[31] D. Prochnow and T. M. Alm. VLDB. 2017.
361 [[32] D. Prochnow and T. M. Alm. VLDB. 2017.
362 [[33] D. Prochnow and T. M. Alm. VLDB. 2017.
363 [[34] D. Prochnow and T. M. Alm. VLDB. 2017.
364 [[35] D. Prochnow and T. M. Alm. VLDB. 2017.
365 [[36] D. Prochnow and T. M. Alm. VLDB. 2017.
366 [[37] D. Prochnow and T. M. Alm. VLDB. 2017.
367 [[38] D. Prochnow and T. M. Alm. VLDB. 2017.
368 [[39] D. Prochnow and T. M. Alm. VLDB. 2017.
369 [[40] D. Prochnow and T. M. Alm. VLDB. 2017.
370 [[41] D. Prochnow and T. M. Alm. VLDB. 2017.
371 [[42] D. Prochnow and T. M. Alm. VLDB. 2017.
372 [[43] D. Prochnow and T. M. Alm. VLDB. 2017.
373 [[44] D. Prochnow and T. M. Alm. VLDB. 2017.
374 [[45] D. Prochnow and T. M. Alm. VLDB. 2017.
375 [[46] D. Prochnow and T. M. Alm. VLDB. 2017.
376 [[47] D. Prochnow and T. M. Alm. VLDB. 2017.
377 [[48] D. Prochnow and T. M. Alm. VLDB. 2017.
378 [[49] D. Prochnow and T. M. Alm. VLDB. 2017.
379 [[50] D. Prochnow and T. M. Alm. VLDB. 2017.
380 [[51] D. Prochnow and T. M. Alm. VLDB. 2017.
381 [[52] D. Prochnow and T. M. Alm. VLDB. 2017.
382 [[53] D. Prochnow and T. M. Alm. VLDB. 2017.
383 [[54] D. Prochnow and T. M. Alm. VLDB. 2017.
384 [[55] D. Prochnow and T. M. Alm. VLDB. 2017.
385 [[56] D. Prochnow and T. M. Alm. VLDB. 2017.
386 [[57] D. Prochnow and T. M. Alm. VLDB. 2017.
387 [[58] D. Prochnow and T. M. Alm. VLDB. 2017.
388 [[59] D. Prochnow and T. M. Alm. VLDB. 2017.
389 [[60] D. Prochnow and T. M. Alm. VLDB. 2017.
390 [[61] D. Prochnow and T. M. Alm. VLDB. 2017.
391 [[62] D. Prochnow and T. M. Alm. VLDB. 2017.
392 [[63] D. Prochnow and T. M. Alm. VLDB. 2017.
393 [[64] D. Prochnow and T. M. Alm. VLDB. 2017.
394 [[65] D. Prochnow and T. M. Alm. VLDB. 2017.
395 [[66] D. Prochnow and T. M. Alm. VLDB. 2017.
396 [[67] D. Prochnow and T. M. Alm. VLDB. 2017.
397 [[68] D. Prochnow and T. M. Alm. VLDB. 2017.
398 [[69] D. Prochnow and T. M. Alm. VLDB. 2017.
399 [[70] D. Prochnow and T. M. Alm. VLDB. 2017.
400 [[71] D. Prochnow and T. M. Alm. VLDB. 2017.
401 [[72] D. Prochnow and T. M. Alm. VLDB. 2017.
402 [[73] D. Prochnow and T. M. Alm. VLDB. 2017.
403 [[74] D. Prochnow and T. M. Alm. VLDB. 2017.
404 [[75] D. Prochnow and T. M. Alm. VLDB. 2017.
405 [[76] D. Prochnow and T. M. Alm. VLDB. 2017.
406 [[77] D. Prochnow and T. M. Alm. VLDB. 2017.
407 [[78] D. Prochnow and T. M. Alm. VLDB. 2017.
408 [[79] D. Prochnow and T. M. Alm. VLDB. 2017.
409 [[80] D. Prochnow and T. M. Alm. VLDB. 2017.
410 [[81] D. Prochnow and T. M. Alm. VLDB. 2017.
411 [[82] D. Prochnow and T. M. Alm. VLDB. 2017.
412 [[83] D. Prochnow and T. M. Alm. VLDB. 2017.
413 [[84] D. Prochnow and T. M. Alm. VLDB. 2017.
414 [[85] D. Prochnow and T. M. Alm. VLDB. 2017.
415 [[86] D. Prochnow and T. M. Alm. VLDB. 2017.
416 [[87] D. Prochnow and T. M. Alm. VLDB. 2017.
417 [[88] D. Prochnow and T. M. Alm. VLDB. 2017.
418 [[89] D. Prochnow and T. M. Alm. VLDB. 2017.
419 [[90] D. Prochnow and T. M. Alm. VLDB. 2017.
420 [[91] D. Prochnow and T. M. Alm. VLDB. 2017.
421 [[92] D. Prochnow and T. M. Alm. VLDB. 2017.
422 [[93] D. Prochnow and T. M. Alm. VLDB. 2017.
423 [[94] D. Prochnow and T. M. Alm. VLDB. 2017.
424 [[95] D. Prochnow and T. M. Alm. VLDB. 2017.
425 [[96] D. Prochnow and T. M. Alm. VLDB. 2017.
426 [[97] D. Prochnow and T. M. Alm. VLDB. 2017.
427 [[98] D. Prochnow and T. M. Alm. VLDB. 2017.
428 [[99] D. Prochnow and T. M. Alm. VLDB. 2017.
429 [[100] D. Prochnow and T. M. Alm. VLDB. 2017.
430 [[101] D. Prochnow and T. M. Alm. VLDB. 2017.
431 [[102] D. Prochnow and T. M. Alm. VLDB. 2017.
432 [[103] D. Prochnow and T. M. Alm. VLDB. 2017.
433 [[104] D. Prochnow and T. M. Alm. VLDB. 2017.
434 [[105] D. Prochnow and T. M. Alm. VLDB. 2017.
435 [[106] D. Prochnow and T. M. Alm. VLDB. 2017.
436 [[107] D. Prochnow and T. M. Alm. VLDB. 2017.
437 [[108] D. Prochnow and T. M. Alm. VLDB. 2017.
438 [[109] D. Prochnow and T. M. Alm. VLDB. 2017.
439 [[110] D. Prochnow and T. M. Alm. VLDB. 2017.
440 [[111] D. Prochnow and T. M. Alm. VLDB. 2017.
441 [[112] D. Prochnow and T. M. Alm. VLDB. 2017.
442 [[113] D. Prochnow and T. M. Alm. VLDB. 2017.
443 [[114] D. Prochnow and T. M. Alm. VLDB. 2017.
444 [[115] D. Prochnow and T. M. Alm. VLDB. 2017.
445 [[116] D. Prochnow and T. M. Alm. VLDB. 2017.
446 [[117] D. Prochnow and T. M. Alm. VLDB. 2017.
447 [[118] D. Prochnow and T. M. Alm. VLDB. 2017.
448 [[119] D. Prochnow and T. M. Alm. VLDB. 2017.
449 [[120] D. Prochnow and T. M. Alm. VLDB. 2017.
450 [[121] D. Prochnow and T. M. Alm. VLDB. 2017.
451 [[122] D. Prochnow and T. M. Alm. VLDB. 2017.
452 [[123] D. Prochnow and T. M. Alm. VLDB. 2017.
453 [[124] D. Prochnow and T. M. Alm. VLDB. 2017.
454 [[125] D. Prochnow and T. M. Alm. VLDB. 2017.
455 [[126] D. Prochnow and T. M. Alm. VLDB. 2017.
456 [[127] D. Prochnow and T. M. Alm. VLDB. 2017.
457 [[128] D. Prochnow and T. M. Alm. VLDB. 2017.
458 [[129] D. Prochnow and T. M. Alm. VLDB. 2017.
459 [[130] D. Prochnow and T. M. Alm. VLDB. 2017.
460 [[131] D. Prochnow and T. M. Alm. VLDB. 2017.
461 [[132] D. Prochnow and T. M. Alm. VLDB. 2017.
462 [[133] D. Prochnow and T. M. Alm. VLDB. 2017.
463 [[134] D. Prochnow and T. M. Alm. VLDB. 2017.
464 [[135] D. Prochnow and T. M. Alm. VLDB. 2017.
465 [[136] D. Prochnow and T. M. Alm. VLDB. 2017.
466 [[137] D. Prochnow and T. M. Alm. VLDB. 2017.
467 [[138] D. Prochnow and T. M. Alm. VLDB. 2017.
468 [[139] D. Prochnow and T. M. Alm. VLDB. 2017.
469 [[140] D. Prochnow and T. M. Alm. VLDB. 2017.
470 [[141] D. Prochnow and T. M. Alm. VLDB. 2017.
471 [[142] D. Prochnow and T. M. Alm. VLDB. 2017.
472 [[143] D. Prochnow and T. M. Alm. VLDB. 2017.
473 [[144] D. Prochnow and T. M. Alm. VLDB. 2017.
474 [[145] D. Prochnow and T. M. Alm. VLDB. 2017.
475 [[146] D. Prochnow and T. M. Alm. VLDB. 2017.
476 [[147] D. Prochnow and T. M. Alm. VLDB. 2017.
477 [[148] D. Prochnow and T. M. Alm. VLDB. 2017.
478 [[149] D. Prochnow and T. M. Alm. VLDB. 2017.
479 [[150] D. Prochnow and T. M. Alm. VLDB. 2017.
480 [[151] D. Prochnow and T. M. Alm. VLDB. 2017.
481 [[152] D. Prochnow and T. M. Alm. VLDB. 2017.
482 [[153] D. Prochnow and T. M. Alm. VLDB. 2017.
483 [[154] D. Prochnow and T. M. Alm. VLDB. 2017.
484 [[155] D. Prochnow and T. M. Alm. VLDB. 2017.
485 [[156] D. Prochnow and T. M. Alm. VLDB. 2017.
486 [[157] D. Prochnow and T. M. Alm. VLDB. 2017.
487 [[158] D. Prochnow and T. M. Alm. VLDB. 2017.
488 [[159] D. Prochnow and T. M. Alm. VLDB. 2017.
489 [[160] D. Prochnow and T. M. Alm. VLDB. 2017.
490 [[161] D. Prochnow and T. M. Alm. VLDB. 2017.
491 [[162] D. Prochnow and T. M. Alm. VLDB. 2017.
492 [[163] D. Prochnow and T. M. Alm. VLDB. 2017.
493 [[164] D. Prochnow and T. M. Alm. VLDB. 2017.
494 [[165] D. Prochnow and T. M. Alm. VLDB. 2017.
495 [[166] D. Prochnow and T. M. Alm. VLDB. 2017.
496 [[167] D. Prochnow and T. M. Alm. VLDB. 2017.
497 [[168] D. Prochnow and T. M. Alm. VLDB. 2017.
498 [[169] D. Prochnow and T. M. Alm. VLDB. 2017.
499 [[170] D. Prochnow and T. M. Alm. VLDB. 2017.
500 [[171] D. Prochnow and T. M. Alm. VLDB. 2017.
501 [[172] D. Prochnow and T. M. Alm. VLDB. 2017.
502 [[173] D. Prochnow and T. M. Alm. VLDB. 2017.
503 [[174] D. Prochnow and T. M. Alm. VLDB. 2017.
504 [[175] D. Prochnow and T. M. Alm. VLDB. 2017.
505 [[176] D. Prochnow and T. M. Alm. VLDB. 2017.
506 [[177] D. Prochnow and T. M. Alm. VLDB. 2017.
507 [[178] D. Prochnow and T. M. Alm. VLDB. 2017.
508 [[179] D. Prochnow and T. M. Alm. VLDB. 2017.
509 [[180] D. Prochnow and T. M. Alm. VLDB. 2017.
510 [[181] D. Prochnow and T. M. Alm. VLDB. 2017.
511 [[182] D. Prochnow and T. M. Alm. VLDB. 2017.
512 [[183] D. Prochnow and T. M. Alm. VLDB. 2017.
513 [[184] D. Prochnow and T. M. Alm. VLDB. 2017.
514 [[185] D. Prochnow and T. M. Alm. VLDB. 2017.
515 [[186] D. Prochnow and T. M. Alm. VLDB. 2017.
516 [[187] D. Prochnow and T. M. Alm. VLDB. 2017.
517 [[188] D. Prochnow and T. M. Alm. VLDB. 2017.
518 [[189] D. Prochnow and T. M. Alm. VLDB. 2017.
519 [[190] D. Prochnow and T. M. Alm. VLDB. 2017.
520 [[191] D. Prochnow and T. M. Alm. VLDB. 2017.
521 [[192] D. Prochnow and T. M. Alm. VLDB. 2017.
522 [[193] D. Prochnow and T. M. Alm. VLDB. 2017.
523 [[194] D. Prochnow and T. M. Alm. VLDB. 2017.
524 [[195] D. Prochnow and T. M. Alm. VLDB. 2017.
525 [[196] D. Prochnow and T. M. Alm. VLDB. 2017.
526 [[197] D. Prochnow and T. M. Alm. VLDB. 2017.
527 [[198] D. Prochnow and T. M. Alm. VLDB. 2017.
528 [[199] D. Prochnow and T. M. Alm. VLDB. 2017.
529 [[200] D. Prochnow and T. M. Alm. VLDB. 2017.
530 [[201] D. Prochnow and T. M. Alm. VLDB. 2017.
531 [[202] D. Prochnow and T. M. Alm. VLDB. 2017.
532 [[203] D. Prochnow and T. M. Alm. VLDB. 2017.
533 [[204] D. Prochnow and T. M. Alm. VLDB. 2017.
534 [[205] D. Prochnow and T. M. Alm. VLDB. 2017.
535 [[206] D. Prochnow and T. M. Alm. VLDB. 2017.
536 [[207] D. Prochnow and T. M. Alm. VLDB. 2017.
537 [[208] D. Prochnow and T. M. Alm. VLDB. 2017.
538 [[209] D. Prochnow and T. M. Alm. VLDB. 2017.
539 [[210] D. Prochnow and T. M. Alm. VLDB. 2017.
540 [[211] D. Prochnow and T. M. Alm. VLDB. 2017.
541 [[212] D. Prochnow and T. M. Alm. VLDB. 2017.
542 [[213] D. Prochnow and T. M. Alm. VLDB. 2017.
543 [[214] D. Prochnow and T. M. Alm. VLDB. 2017.
544 [[215] D. Prochnow and T. M. Alm. VLDB. 2017.
545 [[216] D. Prochnow and T. M. Alm. VLDB. 2017.
546 [[217] D. Prochnow and T. M. Alm. VLDB. 2017.
547 [[218] D. Prochnow and T. M. Alm. VLDB. 2017.
548 [[219] D. Prochnow and T. M. Alm. VLDB. 2017.
549 [[220] D. Prochnow and T. M. Alm. VLDB. 2017.
550 [[221] D. Prochnow and T. M. Alm. VLDB. 2017.
551 [[222] D. Prochnow and T. M. Alm. VLDB. 2017.
552 [[223] D. Prochnow and T. M. Alm. VLDB. 2017.
553 [[224] D. Prochnow and T. M. Alm. VLDB. 2017.
554 [[225] D. Prochnow and T. M. Alm. VLDB. 2017.
555 [[226] D. Prochnow and T. M. Alm. VLDB. 2017.
556 [[227] D. Prochnow and T. M. Alm. VLDB. 2017.
557 [[228] D. Prochnow and T. M. Alm. VLDB. 2017.
558 [[229] D. Prochnow and T. M. Alm. VLDB. 2017.
559 [[230] D. Prochnow and T. M. Alm. VLDB. 2017.
560 [[231] D. Prochnow and T. M. Alm. VLDB. 2017.
561 [[232] D. Prochnow and T. M. Alm. VLDB. 2017.
562 [[233] D. Prochnow and T. M. Alm. VLDB. 2017.
563 [[234] D. Prochnow and T. M. Alm. VLDB. 2017.
564 [[235] D. Prochnow and T. M. Alm. VLDB. 2017.
565 [[236] D. Prochnow and T. M. Alm. VLDB. 2017.
566 [[237] D. Prochnow and T. M. Alm. VLDB. 2017.
567 [[238] D. Prochnow and T. M. Alm. VLDB. 2017.
568 [[239] D. Prochnow and T. M. Alm. VLDB. 2017.
569 [[240] D. Prochnow and T. M. Alm. VLDB. 2017.
570 [[241] D. Prochnow and T. M. Alm. VLDB. 2017.
571 [[242] D. Prochnow and T. M. Alm. VLDB. 2017.
572 [[243] D. Prochnow and T. M. Alm. VLDB. 2017.
573 [[244] D. Prochnow and T. M. Alm. VLDB. 2017.
574 [[245] D. Prochnow and T. M. Alm. VLDB. 2017.
575 [[246] D. Prochnow and T. M. Alm. VLDB. 2017.
576 [[247] D. Prochnow and T. M. Alm. VLDB. 2017.
577 [[248] D. Prochnow and T. M. Alm. VLDB. 2017.
578 [[249] D. Prochnow and T. M. Alm. VLDB. 2017.
579 [[250] D. Prochnow and T. M. Alm. VLDB. 2017.
580 [[251] D. Prochnow and T. M. Alm. VLDB. 2017.
581 [[252] D. Prochnow and T. M. Alm. VLDB. 2017.
582 [[253] D. Prochnow and T. M. Alm. VLDB. 2017.
583 [[254] D. Prochnow and T. M. Alm. VLDB. 2017.
584 [[255] D. Prochnow and T. M. Alm. VLDB. 2017.
585 [[256] D. Prochnow and T. M. Alm. VLDB. 2017.
586 [[257] D. Prochnow and T. M. Alm. VLDB. 2017.
587 [[258] D. Prochnow and T. M. Alm. VLDB. 2017.
588 [[259] D. Prochnow and T. M. Alm. VLDB. 2017.
589 [[260] D. Prochnow and T. M. Alm. VLDB. 2017.
590 [[261] D. Prochnow and T. M. Alm. VLDB. 2017.
591 [[262] D. Prochnow and T. M. Alm. VLDB. 2017.
592 [[263] D. Prochnow and T. M. Alm. VLDB. 2017.
593 [[264] D. Prochnow and T. M. Alm. VLDB. 2017.
594 [[265] D. Prochnow and T. M. Alm. VLDB. 2017.
595 [[266] D. Prochnow and T. M. Alm. VLDB. 2017.
596 [[267] D. Prochnow and T. M. Alm. VLDB. 2017.
597 [[268] D. Prochnow and T. M. Alm. VLDB. 2017.
598 [[269] D. Prochnow and T. M. Alm. VLDB. 2017.
599 [[270] D. Prochnow and T. M. Alm. VLDB. 2017.
600 [[271] D. Prochnow and T. M. Alm. VLDB. 2017.
601 [[272] D. Prochnow and T. M. Alm. VLDB. 2017.
602 [[273] D. Prochnow and T. M. Alm. VLDB. 2017.
603 [[274] D. Prochnow and T. M. Alm. VLDB. 2017.
604 [[275] D. Prochnow and T. M. Alm. VLDB. 2017.
605 [[276] D. Prochnow and T. M. Alm. VLDB. 2017.
606 [[277] D. Prochnow and T. M. Alm. VLDB. 2017.
607 [[278] D. Prochnow and T. M. Alm. VLDB. 2017.
608 [[279] D. Prochnow and T. M. Alm. VLDB. 2017.
609 [[280] D. Prochnow and T. M. Alm. VLDB. 2017.
610 [[281] D. Prochnow and T. M. Alm. VLDB. 2017.
611 [[282] D. Prochnow and T. M. Alm. VLDB. 2017.
612 [[283] D. Prochnow and T. M. Alm. VLDB. 2017.
613 [[284] D. Prochnow and T. M. Alm. VLDB. 2017.
614 [[285] D. Prochnow and T. M. Alm. VLDB. 2017.
615 [[286] D. Prochnow and T. M. Alm. VLDB. 2017.
616 [[287] D. Prochnow and T. M. Alm. VLDB. 2017.
617 [[288] D. Prochnow and T. M. Alm. VLDB. 2017.
618 [[289] D. Prochnow and T. M. Alm. VLDB. 2017.
619 [[290] D. Prochnow and T. M. Alm. VLDB. 2017.
620 [[291] D. Prochnow and T. M. Alm. VLDB. 2017.
621 [[292] D. Prochnow and T. M. Alm. VLDB. 2017.
622 [[293] D. Prochnow and T. M. Alm. VLDB. 2017.
623 [[294] D. Prochnow and T. M. Alm. VLDB. 2017.
624 [[295] D. Prochnow and T. M. Alm. VLDB. 2017.
625 [[296] D. Prochnow and T. M. Alm. VLDB. 2017.
626 [[297] D. Prochnow and T. M. Alm. VLDB. 2017.
627 [[298] D. Prochnow and T. M. Alm. VLDB. 2017.
628 [[299] D. Prochnow and T. M. Alm. VLDB. 2017.
629 [[300] D. Prochnow and T. M. Alm. VLDB. 2017.
630 [[301] D. Prochnow and T. M. Alm. VLDB. 2017.
631 [[302] D. Prochnow and T. M. Alm. VLDB. 2017.
632 [[303] D. Prochnow and T. M. Alm. VLDB. 2017.
633 [[304] D. Prochnow and T. M. Alm. VLDB. 2017.
634 [[305] D. Prochnow and T. M. Alm. VLDB. 2017.
635 [[306] D. Prochnow and T. M. Alm. VLDB. 2017.
636 [[307] D. Prochnow and T. M. Alm. VLDB. 2017.
637 [[308] D. Prochnow and T. M. Alm. VLDB. 2017.
638 [[309] D. Prochnow and T. M. Alm. VLDB. 2017.
639 [[310] D. Prochnow and T. M. Alm. VLDB. 2017.
640 [[311] D. Prochnow and T. M. Alm. VLDB. 2017.
641 [[312] D. Prochnow and T. M. Alm. VLDB. 2017.
642 [[313] D. Prochnow and T. M. Alm. VLDB. 2017.
643 [[314] D. Prochnow and T. M. Alm. VLDB. 2017.
644 [[315] D. Prochnow and T. M. Alm. VLDB. 2017.
645 [[316] D. Prochnow and T. M. Alm. VLDB. 2017.
646 [[317] D. Prochnow and T. M. Alm. VLDB. 2017.
647 [[318] D. Prochnow and T. M. Alm. VLDB. 2017.
648 [[319] D. Prochnow and T. M. Alm. VLDB. 2017.
649 [[320] D. Prochnow and T. M. Alm. VLDB. 2017.
650 [[321] D. Prochnow and T. M. Alm. VLDB. 2017.
651 [[322] D. Prochnow and T. M. Alm. VLDB. 2017.
652 [[323] D. Prochnow and T. M. Alm. VLDB. 2017.
653 [[324] D. Prochnow and T. M. Alm. VLDB. 2017.
654 [[325] D. Prochnow and T. M. Alm. VLDB. 2017.
655 [[326] D. Prochnow and T. M. Alm. VLDB. 2017.
656 [[327] D. Prochnow and T. M. Alm. VLDB. 2017.
657 [[328] D. Prochnow and T. M. Alm. VLDB. 2017.
658 [[329] D. Prochnow and T. M. Alm. VLDB. 2017.
659 [[330] D. Prochnow and T. M. Alm. VLDB. 2017.
660 [[331] D. Prochnow and T. M. Alm. VLDB. 2017.
661 [[332] D. Prochnow and T. M. Alm. VLDB. 2017.
662 [[333] D. Prochnow and T. M. Alm. VLDB. 2017.
663 [[334] D. Prochnow and T. M. Alm. VLDB. 2017.
664 [[335] D. Prochnow and T. M. Alm. VLDB. 2017.
665 [[336] D. Prochnow and T. M. Alm. VLDB. 2017.
666 [[337] D. Prochnow and T. M. Alm. VLDB. 2017.
667 [[338] D. Prochnow and T. M. Alm. VLDB. 2017.
668 [[339] D. Prochnow and T. M. Alm. VLDB. 2017.
669 [[340] D. Prochnow and T. M. Alm. VLDB. 2017.
670 [[341] D. Prochnow and T. M. Alm. VLDB. 2017.
671 [[342] D. Prochnow and T. M. Alm. VLDB. 2017.
672 [[343] D. Prochnow and T. M. Alm. VLDB. 2017.
673 [[344] D. Prochnow and T. M. Alm. VLDB. 2017.
674 [[345] D. Prochnow and T. M. Alm. VLDB. 2017.
675 [[346] D. Prochnow and T. M. Alm. VLDB. 2017.
676 [[347] D. Prochnow and T. M. Al
```

6 CONCLUSION

In this paper, we presented the DocLayNet dataset. It provides the document conversion and layout analysis from the dataset, we have derived on the one hand reference metrics for human performance on document layout.

To date, there is still a significant gap between human and ML accuracy on the layout interpretation task.

REFERENCES

- [#] Max Gobet, Tamir Hassan, Ermelinda Gao, and Giorgio Ori. Icdar 2013 table competition. In 2013
- [#] Christiaan Clausmer, Apostolos Antonacopoulou, and Stefan Peterschach. Icdar2017 competition. 68
- [#] Hervé Jannin, Jean-Luc Mennier, Liangcai Gao, Yilun Huang, Yu Fang, Florian Kleber, and Eva-Mari
- [#] Mingming Jiang, Mingming Jiang, Peter Zandbergen, and Douglas C. Staelens. Competition on scientific literature layout analysis. In ICDAR, pages 1–6, 2017.
- [#] Logan MacLean, Michael Rausch, and Paul N. Gallo. Icdar2019 competition. In ICDAR, pages 1–6, 2019.
- [#] Xu Sheng, Jianbin Yang, and Antonio Jimenez-Vega. Publindex: Largest dataset ever for document layout analysis. In ICDAR, pages 1–6, 2019.
- [#] Minghang Li, Yiheng Xua, Lei Cui, Shexian Huang, Furu Wei, Shoujiao Li, and Min Zhou. Docbank: A book layout dataset. In ICDAR, pages 1–6, 2019.
- [#] Riaz Ahmad, Muhammad Tamim Afzal, and M. Qadir. Information extraction from pdf files based on deep learning. In ICDAR, pages 1–6, 2019.
- [#] Ross B. Girshick. Fast r-cnn. In NIPS, pages 941–949. IEEE International Conference on Computer Vision, 2015.
- [#] Shaogang Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection. In NIPS, pages 911–919. IEEE International Conference on Computer Vision, 2015.
- [#] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In IEEE International Conference on Computer Vision, 2017.
- [#] Glenn Jocher, Alex Stokan, Ayush Chaurasia, Jirka Boček, NanoCode01, TackKie, Yonghie Kwon, K

Figure 6: Example layout predictions on selected pages from the DocLayNet test-set. (A, D) exhibit fav

```
<!-- image -->
```

+Hiicons, Mai Thanh Minh, Marc, albinxav1, fatih, cleg, and wanghai yang. Ultralytics/yolov5: v6.0 - Yo

+ [14] Nicolas Caron, Francisco Massa, Gabriel Synnaeve, Nicolas Houssier, Alexander Kirillov, and Ser

+ [15] Mingxing Tan, Baining Fang, and Quoc V. Le. Efficientdet: Scalable and efficient object detecti

+ [16] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, C. Lawrence Zitnick, James Hay

+ [17] Yunjin Wu, Alexander Kirillov, Francisco Massa, Wan-Ten Lo, and Ross Girshick. Distrigrid, James Hay

+ [18] Nicolas Lathouwer, Cesar Berrouco, Mahkam Leyat, Victor Kursunoglu, Ahmed Hassan, Andre Carv

+ [19] Yiheng Li, Minghang Li, Shexian Huang, Furu Wei, and Ming Shuo. Laymlm: Pre-training

Abbildung 8: Beispielhafter Qualitätsunterschied zwischen ursprünglicher und optimierter Docking-Konfiguration (1).

Die Abweichungen umfassten teilweise ganze Tabellen, die in der ursprünglichen Konfiguration fehlten oder unvollständig waren.

⁹¹ We did not control the document selection with regard to language. The vast majority of documents cont
⁹² To ensure that future benchmarks in the document-layout analysis can be easily compared, we split up D
⁹³ <https://www.boundaryreports.com/>
⁹⁴
⁹⁵ Table 1 shows the overall frequency of documents among the different sets. We ensure that subsets are
⁹⁶ In order to accommodate the different types of models currently in use by the community, we provide Doc
⁹⁷ Despite being cost-inducing and far less scalable than other languages, human annotation has several b
⁹⁸
⁹⁹ The annotation campaign was carried out in four phases. In phase one, we identified and prepared the d
¹⁰⁰
¹⁰¹
¹⁰²
¹⁰³
¹⁰⁴
¹⁰⁵
¹⁰⁶
¹⁰⁷
¹⁰⁸
¹⁰⁹
¹¹⁰
¹¹¹
¹¹²
¹¹³
¹¹⁴
¹¹⁵
¹¹⁶
¹¹⁷
¹¹⁸
¹¹⁹
¹²⁰
¹²¹
¹²²
¹²³
¹²⁴
¹²⁵
¹²⁶
¹²⁷
¹²⁸
¹²⁹
¹³⁰
¹³¹
¹³²
¹³³
¹³⁴
¹³⁵
¹³⁶
¹³⁷
¹³⁸
¹³⁹ the textual content of an element, which goes beyond visual layout recognition, in particular outside p
¹⁴⁰ At first sight, the task of visual document-layout interpretation appears intuitive enough to obtain p
¹⁴¹
¹⁴² Obviously, this is an issue with plausible annotations. For example, examples of plausible but inconsi
¹⁴³ (1) Every list-item is an individual object with class `list-item`. This definition is different
¹⁴⁴ (2) One list-item is prepended with hanging indentation. `list-item` elements can be manipulated as list-
¹⁴⁵ (3) For every Caption, there must be exactly one corresponding Picture or Table.
¹⁴⁶ (4) Connected sub-pictures are grouped together in one Picture object.

- * We did not control the document selection with regard to language. The vast majority of documents contain English.
- * To ensure that future benchmarks in the document-layout analysis community can be easily compared, we provide a few examples.
- * e.g. AAPL from <https://www.sec.gov/edgar/searchedgar/companysearch.html>
- * Table 1 shows the overall frequency and distribution of the labels among the different sets. Important to note that the distribution of labels is not uniform across all sets.
- * In order to accommodate the different types of models currently in use by the community, we provide several different representations of the data.
- * Despite being cost-intense and far less scalable than automation, human annotation has several benefits:
 - # # 4 ANNOTATION CAMPAIGN

The annotation campaign was carried out in four phases. In phase one, we identified and prepared the data. Table 1 presents the DocLayout dataset overview. Along with the frequency of each class label, we present the relative frequency of each class label across Train, Test, and Val sets. The last column provides the triple inter-annotator agreement (MAE) for each class label.

class label	Count	% of Total	% of Total	% of Total	triples inter-annotator MAE
Text	125424	1.00	1.00	1.00	0.00
Caption	22524	2.04	1.77	2.32	84.89
Footnote	6318	0.60	0.31	0.58	83.91
Formula	2506	7.22	1.90	2.96	83.85
List-item	155640	15.19	13.24	13.82	83.80
Page-footer	70878	6.51	5.98	6.00	93.94
Page-header	58022	5.10	6.70	5.06	85.89
Picture	45876	4.21	2.76	5.31	69.71
Section-header	10344	0.80	1.77	0.85	83.85
Table	34733	3.20	2.27	3.60	77.91
Text	510377	45.82	49.28	45.00	84.86
Title	5071	0.47	0.30	0.50	60.72
Total	1107470	941123	99816	64531	82.83

Figure 3: Corpus Conversion Service annotation user interface. The PDF page is shown in the background. The interface includes a toolbar with various icons for document manipulation, a search bar, and a status bar indicating the current page number and total pages. A sidebar on the left lists the document structure, including sections like 'Image' and 'Text'.

We distributed the annotation workload and performed continuous quality controls. Phase one and two results are presented in Section 4.2.

Phase 1: Data selection and preparation. Our inclusion criteria for documents were described in Section 2. Preparation work included uploading and parsing the sourced PDF documents in the Corpus Conversion Service.

Phase 2: Label selection and guideline. We reviewed the collected documents and identified the most common labels.

Figure 4: https://arxiv.org/

the textual content of an element, which goes beyond visual layout recognition, in particular outside tables.

At first sight, the task of visual document-layout interpretation appears intuitive enough to obtain precise results. However, this inconsistency in annotations is not desirable for datasets which are intended to be used for training machine learning models.

 - * (1) Every list-item is associated with individual object instances with class List-item. This definition is consistent with the standard definition of a list-item.
 - * (2) Every list-item is a paragraph with hanging indentation. Browsing elements can qualify as List-item.
 - * (3) For every Caption, there is exactly one corresponding Picture or Table.
 - * (4) Connected sub-pictures are grouped together in one Picture object.

Abbildung 9: Beispielhafter Qualitätsunterschied zwischen ursprünglicher und optimierter Docking-Konfiguration (2).

Die Analyse der Pipeline zeigte dabei klar unterscheidbare Sätze „problematischer“ versus „erfolgreicher“ Parameter.

```

226     params = [
227         "pipeline": "vlm",
228         "from_formats": ["docx", "pptx", "html", "image", "pdf", "asciidoc", "md", "xlsx"],
229         "to_formats": ["md", "json", "html", "text", "doctags"], # Option "html_split_page"
230         "image_export_mode": "placeholder", # Allowed values: "placeholder", "embedded", "referenced". Optional, defaults to embedded
231         "do_ocr": True,
232         "force_ocr": False,
233         "ocr_engine": "easyocr",
234         "ocr_lang": ["en"], # en, fr, de, es
235         "pdf_backend": "dlparse_v4",
236         "table_mode": "accurate",
237         "abort_on_error": False,
238
239         "do_table_structure": True, # default is True
240         "include_images": True, # default is True
241         # "do_code_enrichment": True, # default is False
242         # "do_formula_enrichment": True, # default is False
243         # "do_picture_classification": True, # default is False
244         "do_picture_description": True, # default is False
245         "picture_description_api": None, #"http://localhost:11435/v1/",
246         # "vlm_pipeline_model": "granite3.2-vision:2b",
247         # "vlm_pipeline_model_api": "http://localhost:11434/v1/chat/completions", # vlm_pipeline_model_api,
248

```

Abbildung 10: Auszug der als problematisch identifizierten Docding-Parameterkonfiguration.

```

73     parameters = {
74         "from_formats": ["docx", "pptx", "html", "image", "pdf", "asciidoc", "md", "xlsx"],
75         "to_formats": ["md", "json", "html", "text", "doctags"], # Option "html_split_page"
76         "image_export_mode": "placeholder", # Allowed values: placeholder, embedded, referenced. Optional, defaults to embedded.
77         "do_ocr": True,
78         "force_ocr": False,
79         "ocr_engine": "easyocr",
80         "ocr_lang": ["en"],
81         "pdf_backend": "dlparse_v4",
82         "table_mode": "accurate",
83         "abort_on_error": False,
84         # "do_table_structure": True, # default is True
85         # "include_images": True, # default is True
86         # "do_code_enrichment": True, # default is False
87         # "do_formula_enrichment": True, # default is False
88         # "do_picture_classification": True, # default is False
89         # "do_picture_description": True, # default is False
90         # "picture_description_api": "http://localhost:11434/v1/chat/completions",
91         # "vlm_pipeline_model": "granite3.2-vision:2b",
92         # "vlm_pipeline_model_api": vlm_pipeline_model_api,
93
94     }
95     # "target": "zip",

```

Abbildung 11: Auszug der optimierten Docding-Parameterkonfiguration mit deutlich besseren Ergebnissen.

Eine weitere zentrale Herausforderung war, dass 16 GB VRAM nicht ausreichten, um alle Features von Docding in der Container-Variante `docding-serve` stabil zu betreiben. Dies führte wiederholt zu Endlosschleifen und hängenden Prozessen. Als Gegenmaßnahmen wurde einerseits auf die Container-Version verzichtet und Docding stattdessen direkt aus Python heraus verwendet, andererseits wurde die Ausführung auf die CPU verlagert, um das VRAM-Limit zu umgehen.

Zusätzlich beeinträchtigte der Prozess `cloudcode_cli.exe` in der VSCode-Umgebung durch extremen RAM-Verbrauch im Hintergrund die Ausführung von Docding (bis hin zu „freeze“ und „not started“-Zuständen). Dieses Problem konnte nur durch eine vollständige Deinstallation von `cloudcode_cli.exe` behoben werden.

Schliesslich stellte sich das Parsen von Formeln – sowohl auf CPU als auch auf GPU – als sehr langsam heraus. Ein Verzicht auf die Extraktion der Formeln kam jedoch nicht in Frage, da eine möglichst hohe Qualität der Extrakte angestrebt wurde, um die Gesamtperformance des Systems nicht zu beeinträchtigen. Ein exemplarischer Log-Ausschnitt verdeutlicht die extremen Laufzeiten und den schrittweisen Durchsatz der PDF-Batches:

```
[WindowsPath('C:/Users/ML4SE/Desktop/openspec_demo/configs/data/OpenRAGBenc  
h/pdfs/2411.02951v2.pdf')]  
2025-12-17 19:08:35,249 - INFO - detected formats: [<InputFormat.PDF:  
'pdf'>]  
2025-12-17 19:08:35,259 - INFO - Going to convert document batch...  
2025-12-17 19:08:35,260 - INFO - Processing document 2411.02951v2.pdf  
2025-12-18 01:37:07,514 - INFO - Finished converting document  
2411.02951v2.pdf in 23312.29 sec.  
mpve the source file to the target directory  
2025-12-18 01:37:07,940 - INFO - Processed 1 docs, of which 0 failed and 0  
were partially converted.  
2025-12-18 01:37:07,948 - INFO - Document conversion complete in 203589.20  
seconds. it successfully completed 1 out of 287  
[WindowsPath('C:/Users/ML4SE/Desktop/openspec_demo/configs/data/OpenRAGBenc  
h/pdfs/2411.03001v2.pdf')]  
2025-12-18 01:37:07,968 - INFO - detected formats: [<InputFormat.PDF:  
'pdf'>]  
2025-12-18 01:37:07,972 - INFO - Going to convert document batch...  
2025-12-18 01:37:07,973 - INFO - Processing document 2411.03001v2.pdf  
2025-12-18 14:22:26,866 - INFO - Finished converting document  
2411.03001v2.pdf in 45918.92 sec.  
mpve the source file to the target directory  
2025-12-18 14:22:27,152 - INFO - Processed 1 docs, of which 0 failed and 0  
were partially converted.  
2025-12-18 14:22:27,160 - INFO - Document conversion complete in 249508.41  
seconds. it successfully completed 1 out of 286  
[WindowsPath('C:/Users/ML4SE/Desktop/openspec_demo/configs/data/OpenRAGBenc  
h/pdfs/2411.03166v3.pdf')]  
2025-12-18 14:22:27,193 - INFO - detected formats: [<InputFormat.PDF:  
'pdf'>]  
2025-12-18 14:22:27,201 - INFO - Going to convert document batch...  
2025-12-18 14:22:27,202 - INFO - Processing document 2411.03166v3.pdf  
2025-12-19 03:50:46,515 - INFO - Finished converting document  
2411.03166v3.pdf in 48499.35 sec.  
mpve the source file to the target directory  
2025-12-19 03:50:47,201 - INFO - Processed 1 docs, of which 0 failed and 0  
were partially converted.  
2025-12-19 03:50:47,229 - INFO - Document conversion complete in 298008.48  
seconds. it successfully completed 1 out of 285  
[WindowsPath('C:/Users/ML4SE/Desktop/openspec_demo/configs/data/OpenRAGBenc  
h/pdfs/2411.03257v3.pdf')]  
2025-12-19 03:50:47,249 - INFO - detected formats: [<InputFormat.PDF:  
'pdf'>]  
2025-12-19 03:50:47,257 - INFO - Going to convert document batch...  
2025-12-19 03:50:47,259 - INFO - Processing document 2411.03257v3.pdf  
2025-12-19 23:49:15,094 - INFO - Finished converting document  
2411.03257v3.pdf in 71907.86 sec.  
mpve the source file to the target directory  
2025-12-19 23:49:17,939 - INFO - Processed 1 docs, of which 0 failed and 0  
were partially converted.  
2025-12-19 23:49:18,034 - INFO - Document conversion complete in 369919.29  
seconds. it successfully completed 1 out of 284
```

5.2 LinearRAG

Für LinearRAG wurden umfangreiche Evaluationsläufe auf verschiedenen Datensätzen und mit unterschiedlichen LLM-Backends durchgeführt.

Auf dem Datensatz 2wikimultihop wurden vier Konfigurationen verglichen:

- Mit dem lokalen Modell GPT-OSS-20b wurden 658 Passagen (passage_embedding.parquet), 40 320 Entitäten (entity_embedding.parquet) und 21 206 Sätze (sentence_embedding.parquet) geladen. Die Evaluation ergab eine LLM Accuracy von 0.7350 (735/1000) und eine Contain Accuracy von 0.7210 (721/1000).
- Mit dem Online-Modell gpt-4o-mini wurden bei identischer Datenbasis eine LLM Accuracy von 0.6390 (639/1000) und eine Contain Accuracy von 0.6930 (693/1000) erreicht.
- Mit dem Remote-Modell gemma3:17b ergaben sich bei gleicher Datenbasis eine deutlich niedrigere LLM Accuracy von 0.2400 (240/1000) und eine Contain Accuracy von 0.3510 (351/1000).
- Mit dem Online-Modell gpt-4o resultierten eine LLM Accuracy von 0.5900 (590/1000) und eine Contain Accuracy von 0.7550 (755/1000).

Auf dem Datensatz hotpotqa wurden 1 311 Passagen, 66 846 Entitäten und 38 455 Sätze geladen. Mit dem lokalen Modell GPT-OSS-20b ergab sich eine LLM Accuracy von 0.7710 (771/1000) und eine Contain Accuracy von 0.6620 (662/1000).

Auf dem Datensatz musique wurden 1 354 Passagen, 67 532 Entitäten und 39 110 Sätze verarbeitet. Mit GPT-OSS-20b wurden eine LLM Accuracy von 0.6420 (642/1000) und eine Contain Accuracy von 0.3170 (317/1000) erzielt.

Auf dem Datensatz medical wurden 225 Passagen, 9 033 Entitäten und 8 985 Sätze verarbeitet. Die Retrieval-Phase umfasste 2 062 Anfragen. Mit dem lokalen Modell GPT-OSS-20b ergab sich eine LLM Accuracy von 0.6940 (1431/2062) bei einer deutlich niedrigeren Contain Accuracy von 0.0320 (66/2062), was auf einen starken Einfluss des im Modell bereits vorhandenen domänspezifischen Wissens bei gleichzeitig begrenzter Evidenzabdeckung im Retrieval hinweist.

5.3 GraphMERT

!TODO:

5.4 LeanRAG

Für LeanRAG wurde insbesondere der Ressourcenbedarf nach einem Refactoring des Schritts der Triple-Extraktion analysiert. Die entsprechende Auswertung zeigt den Verbrauch an CPU-, Speicher- und GPU-Ressourcen während dieses kritischen Verarbeitungsschritts.

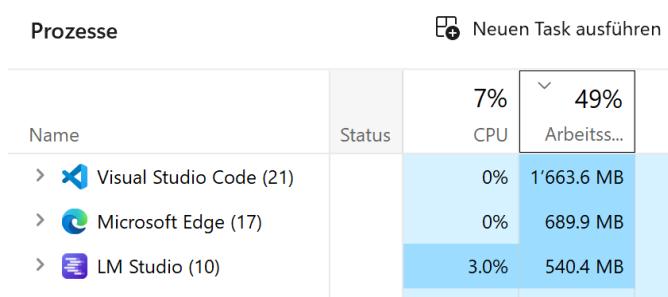


Abbildung 12: Ressourcenbedarf im LeanRAG-Pipeline-Schritt der Triple-Extraktion nach dem Refactoring.

5.5 Benchmark

6. Diskussion

6.2 LinearRAG

Für die Evaluierung mittels OpenRAGBench wurde ein spezifischer Graph basierend auf einem Korpus von **1001 wissenschaftlichen Publikationen** (Arxiv) erstellt. Die Graph-Konstruktion erfolgte vollständig deterministisch unter Verwendung des **scispacy** Modells, ohne die Verwendung von LLM-Token für die Extraktion. Als Evaluator kam der **TRECEvaluator** in Kombination mit **Gemini-2.5-flash** zum Einsatz, wie in der Konfiguration definiert.

Die nachfolgende Tabelle fasst die Metriken des erstellten Linear RAG Graphen zusammen und verdeutlicht die Skalierbarkeit des Ansatzes:

Metrik	Wert	Beschreibung
Rohdaten		
Anzahl Dokumente (Papers)	1,001	Korpusgrösse
Verarbeitetes Datenvolumen	25.6 MB	Raw text bytes
Passages (Chunks)	278,692	Erzeugte Textabschnitte
Extraktion (scispacy)		
Extrahierte Sätze	1,198,328	Identifizierte Sentence Units
Extrahierte Entitäten (Total)	3,650,438	Inkl. Duplikate über alle Chunks
Eindeutige Entitäten	596,824	Unique Nodes im Graphen
Graph Topologie		
Graph Nodes (Total)	1,751,262	Summe aus Passage, Sentence & Entity Nodes
Graph Edges (Total)	7,370,454	Strukturelle & statistische Verbindungen

Die hohe Anzahl an Kanten (über 7.3 Millionen) im Verhältnis zu den Knoten zeigt die hohe Dichte der Vernetzung, die durch den algorithmischen Ansatz ("Relation-free") erreicht wurde. Bemerkenswert ist, dass trotz der Extraktion von über 3.6 Millionen Entitäten die Verarbeitung rein CPU-basiert und effizient erfolgte.

6.3 GraphMERT

6.4 LeanRAG

6.5 Benchmark

!TODO:

7. Conclusion / Fazit

Das Projekt IMARA hatte das Ziel, eine domänenspezifische GraphRAG-Pipeline mit Modell-Fine-tuning vorzubereiten und die Effektivität graphbasierter RAG-Ansätze im Vergleich zu naivem RAG zu evaluieren.

Zentrale Ergebnisse:

- Es wurde eine skalierbare LinearRAG-Implementierung realisiert, die auf einem grossen wissenschaftlichen Korpus (OpenRAGBench) einen dichten Wissensgraphen mit über 1.7 Mio. Knoten und 7.3 Mio. Kanten aufbaut.
- Die Evaluierung zeigt, dass LinearRAG auf Multi-Hop-Benchmarks konkurrenzfähige bis sehr gute Genauigkeiten erzielt und klassische Limitierungen vektorbasierter RAG-Systeme adressiert.
- Die Arbeit bestätigt, dass Datenqualität (insbesondere PDF-Extraktion) und Graphdesign entscheidende Hebel für die Gesamtleistung sind.
- Durch DVC, MLflow und eine modulare Architektur ist die Grundlage für reproduzierbare Experimente und zukünftiges Fine-tuning gelegt.

Insgesamt konnte das Kernziel erreicht werden: Der Nutzen graphbasierter RAG-Ansätze gegenüber naiven Vektor-RAGs wurde qualitativ und quantitativ demonstriert. Gleichzeitig wurden offene Fragen und Herausforderungen klar sichtbar gemacht.

7.1 Persönliches Fazit

7.1.1 Marco Allenspach

!TODO: @Marco Allenspach

7.1.2 Lukas Koller

!TODO: @Lukas Koller

7.1.3 Emanuel Sovrano

!TODO: @Emanuel Sovrano

-
- Der Vorsatz Plattformunabhängig zu sein hatte sich im Laufe des Projekts als unnötige Herausforderung herausgestellt. Konkret Microsoft Windows hatte bei der Installation spezielle Anforderungen, Inkompatibilität mit MLFlow und letztlich erzwungene Reboots, die mehrfach lang laufende Prozesse abgeschossen haben.

8. Risikomanagement und Lessons Learned

8.1 Identifizierte Risiken

Im Projektverlauf wurden mehrere zentrale Risiken sichtbar. Die Datenqualität und die resultierende Graphdichte sind kritisch: Fehler in der PDF-Extraktion oder Entitätserkennung schlagen direkt auf die Qualität des Wissensgraphen und damit auf die Antwortqualität durch. Zudem ist die gesamte Pipeline – von der Docing-Extraktion über die Graphkonstruktion bis hin zu den LLM-Evaluierungen – deutlich rechenintensiv und stellt hohe Anforderungen an Hardware und Laufzeiten. Hinzu kommen Tooling- und Plattformabhängigkeiten: Unterschiedliche Betriebssysteme (Windows, Linux, macOS) sowie Werkzeuge wie VSCode, Supabase, Docing und die zugrunde liegenden Datenbanken und LLM-Backends bringen jeweils ihre eigenen Stolpersteine mit. Schliesslich erhöht die Vielzahl von Komponenten (DVC, MLflow, Supabase, PostgreSQL mit pgvector, LLM-Backends usw.) die Komplexität der Gesamtinfrastruktur und damit den Integrationsaufwand.

8.2 Konkrete Erfahrungen

Die Entscheidung, plattformunabhängig zu bleiben, erwies sich im Alltag als zusätzliche Herausforderung. Insbesondere unter Windows traten Inkompatibilitäten (beispielsweise mit MLflow) und erzwungene Reboots auf, die lang laufende Prozesse – etwa Docling-Batches – wiederholt unterbrachen. Unerwartete Hintergrundprozesse wie `cloudcode_cli.exe` konnten unbemerkt erhebliche Ressourcen blockieren und ML-Workloads so weit beeinträchtigen, dass Konvertierungen nicht mehr starteten oder einfroren. Diese Erfahrungen haben gezeigt, dass eine klare Trennung von produktionsnahen Experimenten und „normalen“ Entwicklungsumgebungen sinnvoll ist; dedizierte Maschinen für rechenintensive Aufgaben wie die Docling-Extraktion vereinfachen Stabilität und Fehlersuche deutlich.

8.3 Lessons Learned

Aus diesen Erfahrungen lassen sich mehrere Lehren ableiten. Es ist hilfreich, möglichst früh im Projekt einen Ende-zu-Ende-Slice zu realisieren – etwa vom PDF bis zur einfachen Antwort –, um Risiken in Tooling, Infrastruktur und Datenflüssen sichtbar zu machen, bevor die Architektur zu komplex wird. Daten- und Modellversionierung sollten von Anfang an konsequent mit Werkzeugen wie DVC und MLflow umgesetzt werden, um Experimente nachvollziehbar, reproduzierbar und vergleichbar zu machen. Schliesslich hat sich gezeigt, dass eine schrittweise Komplexitätssteigerung sinnvoll ist: Zuerst eine stabile, naive RAG-Baseline etablieren, diese messen und verstehen, und darauf aufbauend GraphRAG-Komponenten sowie Fine-tuning iterativ ergänzen, statt alles gleichzeitig zu implementieren.

9. Ausblick

Aus den bisherigen Ergebnissen und Erfahrungen ergeben sich mehrere konkrete Ansatzpunkte für zukünftige Arbeiten. Zunächst bleibt die zentrale Frage, ob ein kompaktes, domänen spezifisches Modell mit rund 80 M Parametern tatsächlich grössere, generische Modelle übertreffen kann. Die bisherigen Experimente deuten darauf hin, dass dies in klar abgegrenzten Domänen und bei gut kuratierten Trainingsdaten möglich ist, allerdings ist dafür eine sehr hohe Qualität der zugrunde liegenden Wissensrepräsentation erforderlich. Ein zentrales Lernfeld ist deshalb die Gestaltung und Pflege des Wissensgraphen selbst.

Die Qualität eines Wissensgraphen wird wesentlich durch die Qualität der Entitäten bestimmt. Sprachliche Mehrdeutigkeiten – etwa Berufsbezeichnungen versus Eigennamen – sollten systematisch durch kontextbasierte Attribute aufgelöst werden. Das Beispiel „Der Müller hat den Beruf eines Maurers“ illustriert dies: Die Entity „Müller“ ist hier kein Beruf, sondern eine Person mit dem Familiennamen Müller und dem Beruf Maurer. Entsprechende Kontexteigenschaften (Rollen, Typen, semantische Klassen) müssen explizit modelliert werden, um Fehlinterpretationen im Graphen zu vermeiden.

Ein weiterer Ansatzpunkt betrifft die Organisation der Verarbeitungsschritte. Möglichst viele Schritte – bis hin zu Entity-Relation-Triples – sollten im Scope eines einzelnen Dokuments vorverarbeitet werden. Dadurch entsteht ein kontinuierlich erweiterbarer Graph, der sich sukzessive aus vorverarbeiteten Datensätzen speist. Gleichzeitig werden Parallelisierung und Lastverteilung erleichtert, und das Entfernen oder Ersetzen einzelner Dokumente wird deutlich einfacher, etwa wenn Daten fehlerhaft sind oder veraltet und durch aktuellere Versionen ersetzt werden müssen. Darüber hinaus ermöglicht dieses Vorgehen, mehrere Graphvarianten mit minimalem Offset für unterschiedliche Berechtigungsstufen zu erzeugen.

Viele Fakten sind zudem orts- und zeitabhängig. Medizinische Interpretationen (z.B. Atemschwierigkeiten auf Meereshöhe versus in grosser Höhe), Aktienkurse oder auch Geschwindigkeitsbegriffe („rasend schnell“)

um 1900 vs. Durchschnitt heute) verändern ihre Bedeutung in Abhängigkeit von Raum und Zeit. Diese Dimensionen sollten systematisch im Graphmodell berücksichtigt werden, etwa durch explizite Zeit- und Ortsattribute oder entsprechende Kontext-Knoten, um Aussagen korrekt einordnen und zeitliche Entwicklungen abbilden zu können.

Für mehrsprachige Knowledge Bases ist eine semantisch korrekte Zusammenführung von Entitäten über Sprachgrenzen hinweg erforderlich. Statt Entitäten lediglich über Übersetzungen zu verknüpfen, bietet sich die Einführung einer höheren Hierarchieschicht in Form eines Konzeptgraphen an: Einzelne Fakten werden in sprachspezifischen Wissensgraphen modelliert und darüber auf sprachunabhängige Konzepte abgebildet. So lassen sich Mehrsprachigkeit und Domänenpezifik besser trennen, ohne permanente Übersetzungen zwischen Sprachen erzwingen zu müssen.

Ein weiterer, vielversprechender Ansatz liegt im Einsatz von Hypergraphen und Relation-Clustering. Das Clustering identischer oder stark ähnlicher Relationen in einem Hypergraphen erlaubt es, Teilgraphen zusammenzuführen, ohne die Möglichkeit zu verlieren, einzelne Teile bei Bedarf wieder zu entfernen. Gleichzeitig können aus den Hyperkanten wahrscheinliche Relationen abgeleitet und zurück in den klassischen Wissensgraph projiziert werden. Dies eröffnet Spielräume für effizienteres Speichern, besseres Generalisieren und für die Ableitung neuer, plausibler Verbindungen.

Schliesslich ist die tiefere Integration der Evaluierungsinfrastruktur ein wichtiger nächster Schritt. Die Ausführung von OpenRAG-Eval-Szenarien soll vollständig über DVC-Pipelines orchestriert werden, sodass Datendownload, Graphaufbau, Retrieval-Läufe und Auswertung automatisch miteinander verknüpft sind und als reproduzierbare Pipelines ausgeführt werden können. Die geplante Arbeit zu DVC und dem Vergleich verschiedener OpenRAG-Eval-Konfigurationen kann hier andocken, indem unterschiedliche Modelle, Konfigurationen und Datenschnitte als DVC-Stages abgebildet und systematisch miteinander verglichen werden. Dies würde die Nachvollziehbarkeit der Experimente weiter erhöhen und eine belastbare Grundlage für die Frage schaffen, unter welchen Bedingungen ein kompaktes, domänenpezifisches Modell grössere, generische LLMs tatsächlich übertreffen kann.

Glossar

A – C

- **AI-Native GraphRAG:** Ein weiterentwickeltes Paradigma von GraphRAG, das den gesamten Workflow von unstrukturierten Daten bis zur Antwortgenerierung automatisiert und dabei die Komplexität von Graphentheorie und Datenbankmanagement abstrahiert.
- **Chunking:** Der Prozess des Zerlegens von Texten in kleinere Abschnitte (Chunks). Im Bericht wird dies als kritischer Faktor für naives RAG identifiziert, da suboptimale Chunk-Größen (zu gross oder zu klein) zu Kontextverlust oder Rauschen führen können.
- **CommonKG:** Eine im Kontext von LeanRAG erwähnte Methode zur Erstellung von Wissensgraphen, bei der Entitäten und Relationen (Triples) aus Text-Chunks extrahiert und dedupliziert werden.

D – G

- **Docling:** Ein Open-Source-Toolkit zur Dokumentenkonvertierung, das im Projekt eingesetzt wurde, um komplexe PDFs in maschinenlesbare Formate (z.B. Markdown, JSON, Doctags) zu wandeln. Im Projekt traten Herausforderungen bezüglich VRAM-Verbrauch, Laufzeit und Stabilität (z.B. in der Container-Variante `docling-serve`) auf.

- **Embeddings:** Vektorrepräsentationen von Texten (z.B. Sätze, Entitäten, Passagen), die als Grundlage für Vektorschre, Clustering und Ähnlichkeitsberechnungen in RAG- und GraphRAG-Systemen dienen.
- **FActScore / ValidityScore:** Metriken zur Bewertung der faktischen Korrektheit (FActScore) und der ontologischen Gültigkeit von Relationen (ValidityScore) in Wissensgraphen oder generierten Antworten. Im Bericht erzielt GraphMERT deutlich höhere Werte als reine LLMs.
- **Fine-tuning:** Das Nachtrainieren eines LLMs (z.B. Qwen) auf spezifischen, oft graphbasierten oder domänenspezifischen Daten, um Antwortqualität und Domänenexpertise zu erhöhen.
- **GraphMERT:** Ein kompaktes, rein grafisches Encoder-Modell (neurosymbolische KI), das effizient zuverlässige und ontologiekonsistente Wissensgraphen aus unstrukturierten Texten generiert und dabei hohe FActScore- und ValidityScore-Werte erreicht.
- **GraphRAG (Graph Retrieval-Augmented Generation):** Eine Erweiterung von RAG, die statt flacher Textlisten strukturierte Wissensgraphen nutzt. Dies ermöglicht das Erkennen komplexer Beziehungen, Multi-Hop-Reasoning und eine explizite Repräsentation von Entitäten und Relationen.
- **Graph-Sparsität:** Ein Mass für die Anzahl fehlender Kanten in einem Graphen relativ zur maximal möglichen Anzahl. Die Sparsität wird typischerweise berechnet als $Sparsität = 1 - (Actual\ Edges / Possible\ Edges)$. Bei 100 % Sparsität existieren keine Kanten, bei 0 % sind alle möglichen Kanten vorhanden. Im Kontext von Wissensgraphen beschreibt sie, wie dicht oder dünn ein Graph verknüpft ist.

H – L

- **Hypergraph:** Eine Graphstruktur, bei der eine Kante (Hyperedge) mehr als zwei Knoten verbinden kann. Im Ausblick wird dies als Ansatz vorgeschlagen, um identische Relationen zu clustern und Teilgraphen effizient zusammenzufassen.
- **IMARA:** Der Name des Projekts. Es steht für die Entwicklung einer domänenspezifischen, graphbasierten RAG-Pipeline mit Modell-Fine-tuning und integrierter Evaluationspipeline.
- **Knowledge Graph (Wissensgraph):** Eine strukturierte Darstellung von Wissen in Form von Knoten (Entitäten) und Kanten (Beziehungen), die ein aktives, abfragefähiges Modell eines Fachbereichs oder der Welt bildet.
- **LeanRAG:** Ein GraphRAG-Ansatz, der auf semantische Aggregation und hierarchisches Retrieval setzt, um Redundanzen zu minimieren (im Bericht ca. 46 % weniger Redundanz im Vergleich zu flachen Baselines) und gleichzeitig hoch relevante Evidenz bereitzustellen.
- **LinearRAG:** Eine effiziente, relation-freie GraphRAG-Methode mit linearer Komplexität. Sie nutzt leichtgewichtige Entity Recognition und semantische Verlinkung zur Graphkonstruktion, ohne LLM-basierte Relationsextraktion, und kombiniert Vektorschre mit graphbasiertem Scoring (z.B. Personalized PageRank).
- **LLM (Large Language Model):** Grosse Sprachmodelle (z.B. GPT-4o, Qwen, Gemma), die als generative Komponente in RAG- und GraphRAG-Pipelines eingesetzt werden, um natürlichsprachliche Antworten aus bereitgestelltem Kontext zu erzeugen.

M – O

- **Multi-Hop-Reasoning:** Die Fähigkeit, Informationen über mehrere Verbindungsschritte hinweg zu verknüpfen (z.B. A ist verbunden mit B, B ist verbunden mit C → Schlussfolgerung von A auf C). Dies ist eine Schwäche von naivem RAG, aber eine Stärke von GraphRAG-Ansätzen wie LeanRAG oder LinearRAG.
- **Naives RAG:** Bezeichnet im Bericht konventionelle, rein vektorbasierte RAG-Architekturen, die Wissen als unzusammenhängende Fakten (Chunks) behandeln, stark von der Chunking-Strategie

abhangen und häufig an kontextueller Fragmentierung leiden.

- **Neurosymbolische KI:** Kombination aus neuronalen Netzwerken (für Generalisierung und Lernen) und symbolischer KI (für Abstraktion, Logik und Graphstrukturen), wie sie im GraphMERT-Ansatz umgesetzt ist.
- **OpenRAGBench:** Ein Referenzdatensatz (Benchmark) mit rund 1000 wissenschaftlichen PDFs (Arxiv) und zugehörigen Frage-Antwort-Paaren, der im Projekt genutzt wird, um die Messbarkeit und Vergleichbarkeit der Ergebnisse sicherzustellen.
- **OpenRAG-Eval:** Ein Evaluations-Framework, das unterschiedliche RAG- und GraphRAG-Systeme anhand einheitlicher Metriken (z.B. LLM Accuracy, Contain Accuracy, Faithfulness, Laufzeit) miteinander vergleicht und im Projekt zur Orchestrierung der Benchmarks eingesetzt wird.

S – V

- **Semantic Aggregation:** Ein Feature von LeanRAG, bei dem Entitäten in semantisch kohärente Zusammenfassungen (Cluster) gruppiert und als aggregierte Knoten mit expliziten Relationen dargestellt werden, um die Navigation im Graphen und das hierarchische Retrieval zu verbessern.
- **Synthetic Data Generation (SDG):** Ein Ansatz zur Generierung künstlicher Testdaten zur Systembewertung, häufig unter Nutzung von LLMs als „Judge“, um z.B. Antwortqualität oder Robustheit zu messen.
- **TF-IDF (Term Frequency-Inverse Document Frequency):** Ein statistisches Mass zur Bewertung der Wichtigkeit eines Terms in einem Dokument relativ zu einer Dokumentensammlung. Eine gebräuchliche Formel ist

$$TF-IDF = \log(1 + tf) \times \log(N / df),$$
wobei tf die Termfrequenz, N die Gesamtzahl der Dokumente und df die Anzahl der Dokumente ist, die den Term enthalten. Im LinearRAG-Ansatz wird TF-IDF unter anderem zur Gewichtung von Passage-Entitäts-Kanten verwendet, um die semantische Relevanz von Entitäten für spezifische Textabschnitte zu quantifizieren.
- **Triple:** Die grundlegende Dateneinheit eines Wissensgraphen, bestehend aus Subjekt, Prädikat (Relation) und Objekt (z.B. „Müller“ → „hat Beruf“ → „Maurer“).
- **Unsloth:** Ein Framework für ressourceneffizientes Fine-tuning von LLMs, das im Projekt genutzt wurde, um Modellanpassungen mit geringeren Hardwareanforderungen durchzuführen.
- **Vektorsimilaritätssuche:** Das Standard-Suchverfahren klassischer RAG-Systeme, bei dem Textabschnitte als Vektoren im Embedding-Raum repräsentiert und basierend auf ihrer Distanz (z.B. Kosinus- oder euklidische Distanz) verglichen werden. Es ermöglicht semantische Suche, berücksichtigt jedoch explizite Relationen zwischen Entitäten oft nicht.

Abbildungsverzeichnis

Abbildung 3: LeanRAG-Framework, übernommen aus Zhang et al. (2025), arXiv:2508.10391. Abbildung 4: LinearRAG-Workflow, übernommen aus Li et al. (2025), arXiv:2510.10114. Abbildung 5: GraphMERT Node Embeddings (t-SNE View) und GraphMERT Semantic Graph Visualization, jeweils übernommen aus Belova et al. (2025), arXiv:2510.09580.

Abbildung 6: Query-Suche auf den Graph-Ergebnissen: links ein perfektes, rechts ein fast perfektes Resultat, übernommen aus Belova et al. (2025), arXiv:2510.09580.

Abbildung 7: Architekturübersicht von Docling, übernommen aus der offiziellen Docling-Dokumentation (Docling-Projekt, Zugriff am 18.01.2026). Abbildung 8: Beispielhafter Qualitätsunterschied zwischen ursprünglicher und optimierter Docling-Konfiguration (1).

Abbildung 9: Beispielhafter Qualitätsunterschied zwischen ursprünglicher und optimierter Docling-

Konfiguration (2).

Abbildung 10: Auszug der als problematisch identifizierten Docling-Parameterkonfiguration.

Abbildung 11: Auszug der optimierten Docling-Parameterkonfiguration mit deutlich besseren Ergebnissen.

Abbildung 12: Ressourcenbedarf im LeanRAG-Pipeline-Schritt der Triple-Extraktion nach dem Refactoring.

Literaturverzeichnis

[1] Docling-Projekt. *Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion.*

arXiv:2501.17887.

Verfügbar unter: <https://www.docling.ai/> (Zugriff am 18.01.2026).

[2] Docling-Projekt (o.J.). *Docling Architecture.*

Verfügbar unter: <https://docling-project.github.io/docling/concepts/architecture/>

(Zugriff am 18.01.2026).

[3] Zhang, X. et al. (2025). *Knowledge-Graph-Based Generation with Semantic Aggregation and Hierarchical Retrieval.*

arXiv:2508.10391.

Zusätzliche Ressourcen: <https://github.com/KnowledgeXLab/LeanRAG>

[4] Li, Y. et al. (2025). *LinearRAG: Linear Graph Retrieval-Augmented Generation on Large-scale Corpora.*

arXiv:2510.10114.

Zusätzliche Ressourcen: <https://github.com/DEEP-PolyU/LinearRAG>

[5] Belova, M., Xiao, J., Tuli, S., & Jha, N. K. (2025). *GraphMERT: Efficient and Scalable Distillation of Reliable Knowledge Graphs from Unstructured Data.*

arXiv:2510.09580.

Zusätzliche Ressourcen: <https://github.com/creativeautomaton/graphMERT-python>

[6] Vectara. *Open RAG Benchmark (1000 PDFs, 3000 Queries): A Multimodal PDF Dataset for Comprehensive RAG Evaluation.*

Verfügbar unter: <https://github.com/vectara/open-rag-bench>