

Projektbericht: IMARA

Domain-specific GraphRAG pipeline with model fine-tuning

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

Datum: [Aktuelles Datum]

Autoren: Marco Allenspach, Lukas Koller, Emanuel Sovrano

Abstract

Die Einführung von Retrieval-Augmented Generation (RAG) markierte einen bedeutenden Meilenstein in der Anwendung grosser Sprachmodelle (LLM), indem generative Fähigkeiten auf faktischen, externen Daten basierten, um Fehlinterpretationen zu vermeiden und die Relevanz zu erhöhen. Um die Schwächen von RAG der ersten Generation durch die Einführung strukturierter, relationaler Kontexte zu beheben, hat sich jedoch mit AI-Native GraphRAG ein weiterentwickeltes Paradigma etabliert.

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. Management Summary

(Ca. 0.5 - 1 Seite) Zusammenfassung des gesamten Projekts: Problemstellung (Extraktion aus komplexen PDFs), gewählter Lösungsansatz (GraphRAG & Fine-tuning) und die wichtigsten Ergebnisse des Benchmarkings.

Traditionelle neuronale Netze eignen sich gut zur Kodierung linearer Beziehungen, doch Daten aus der realen Welt sind in der Regel komplex und multidimensional. Graphen sind besser geeignet, höherdimensionale Verbindungen darzustellen, in denen jeder Knoten mit jedem anderen Knoten in Beziehung steht. Dadurch eignen sich Graphen besser zur Speicherung komplexer Beziehungen aus der realen Welt.

2. Einleitung und Zielsetzung

Das IMARA-Projekt hat zum Ziel, aufzuzeigen wie die Genauigkeit der Abfrage eines graph-basierten RAG-Systems sich verbessert.

Um eine Grundlage für die Messbarkeit zu haben, wurde OpenRAGBench als Referenzdatensatz ausgewählt.

Defining the "AI-Native" GraphRAG Paradigm

AI-Native GraphRAG represents a specific and powerful subset of graph-based RAG systems. Solutions must automate the entire workflow from unstructured data to a natural language answer, abstracting complexities of graph theory and database management.

naives RAG

Die inhärenten Einschränkungen von vektorbasierter RAG

Konventionelle RAG-Architekturen verlassen sich auf Vektorsimilaritätssuche über ein Korpus von geteiltem Text. Dieser Ansatz behandelt Wissen als eine Sammlung von unzusammenhängenden Fakten und hat Schwierigkeiten mit Fragen, die erfordern:

- Synthese von Informationen aus mehreren Quellen
- Verständnis nuancierter Beziehungen zwischen Entitäten
- Durchführung von Multi-Hop-Reasoning Der Kontext, der dem LLM bereitgestellt wird, ist oft eine Liste von Textausschnitten, die keine explizite Darstellung ihrer Verbindungen enthalten.

Kontextuelle Fragmentierung und Blindheit

Das Chunking bricht den natürlichen Informationsfluss willkürlich. Relevanter Kontext kann über verschiedene Chunks, Dokumente oder Abschnitte verstreut sein. Die Vektorschre, 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 Vektorschre ist unwahrscheinlich, diese Informationssprünge zu überbrücken.

Analogie: Vektorbasierte RAG bietet einem Forscher einen Stapel isolierter Karteikarten, während GraphRAG darauf abzielt, eine umfassende Mindmap zu erstellen und bereitzustellen, die entscheidende Verbindungen aufdeckt.

2.1 Projekttitel: IMARA

2.2 Problemstellung

Die Extraktion und Verarbeitung von Informationen aus unstrukturierten PDF-Dokumenten stellt eine Herausforderung für herkömmliche RAG-Systeme dar.

2.3 Projektziele

- Die Implementation von graphbasierten System und der Vergleich zu klassischen RAG-Systemen
- Der Vergleich zwischen verschiedenen graphbasierten RAG-Systemen
-
-

Graph-basiertes RAG: Aufbau einer Pipeline zur Erstellung dichter Wissensgraphen.

•

Model Fine-tuning: Optimierung eines LLMs (z.B. Qwen) basierend auf dem Graph.

•

Automation: End-to-End Automatisierung der Pipeline. Eine flexible Pipeline bauen, die bei der Evaluation der verschiedenen RAG-Systeme unterstützt.

3. Datenbasis und Vorverarbeitung

3.1 Datenquellen

Beschreibung der verwendeten Datensätze, wie z.B. der **Open RAG Bench Dataset** (Arxiv-Kategorien) oder **PubMedQA**.

3.2 PDF-Extraktion mit Docling

Einsatz des **Docling Toolkits** zur effizienten Konvertierung von Dokumenten in maschinenlesbare Formate (Markdown/JSON).

angetroffene Herausforderungen **Challenge:** Die Qualität der Ergebnisse liegt unter den Erwartungen.

Massnahme 1: Optimierung der Parameter. Die optimierte Version der Parameter ist massiv schneller und viel genauer.

```

310  ## 6 CONCLUSION
311
312 #In this paper, we presented the DocLayNet dataset. It provides the document conversion and layout anal
313 #ysis. 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 large dataset for training and testing document layout analysis models.
315
316 To date, there is still a significant gap between human and ML accuracy on the layout interpretation t
317
318 ## REFERENCES
319
320 #\[1\] Max Gobel, Tsvit Hassen, Koenraad Oors, and Giorgio Orsi. Icdl 2011 table competition. In 2011 I
321 #cdl Christian Claeysen, Apostolos Antonopoulos, and Stefan Fleischmann. Research presentation at
322 #\\[2\\] Antonio Cesar, Juan L. Malleino, and Eric G. Horvitz. Icdar 2011 data-based comparison of doc
323 #\\\[3\\\] Antonio Jijon, Peter Zeng, and Douglas S. Hersh. Scientific literature in the International Co
324 #\\\\[4\\\\] Logan Markewich, Hao Yuan, Xin Bing, Xu Li, Sheng, and Jian Li. Data-sharpening for Ime
325 #\\\\\[5\\\\\] Xu Jian, Zhang Bing, and Jian Li. Data-Sharpening for Image Annotation and Document Analysis. In 2011
326 #\\\\\\[6\\\\\\] Ross B. Girshick, Andrew Donahue, Trevor Darrell, and J. Bhattacharya. Technical Journal. Rich feature hierarc
327 #\\\\\\\[7\\\\\\\] Ross B. Girshick, Jason峨峨, and J. Bhattacharya. Technical Journal. International Conference on Com
328 #\\\\\\\\[8\\\\\\\\] Shiqiang Ren, Ross B. Girshick, and J. Bhattacharya. Technical Journal. Fast-recon-towards-the-world
329 #\\\\\\\\\[9\\\\\\\\\] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross B. Girshick. Technical Journal. IEEE Internati
330 #\\\\\\\\\\[10\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
331 #\\\\\\\\\\\[11\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
332 #\\\\\\\\\\\\[12\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
333 #\\\\\\\\\\\\\[13\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
334 #\\\\\\\\\\\\\\[14\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
335 #\\\\\\\\\\\\\\\[15\\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
336 <!-- image -->
337
338 #Biaocong, Mai Thanh Minh, Marc, albinkev, fatih, oleg, and wanghai yang. Ultralytics/yolov5: v6.0 - Yo
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354

```

```

309  ## 6 CONCLUSION
310
311 #In this paper, we presented the DocLayNet dataset. It provides the document conversion and layout anal
312 #ysis. From the dataset, we have derived on the one hand reference metrics for human performance on document
313 #interpretation and on the other hand a large dataset for training and testing document layout analysis models.
314
315 ## REFERENCES
316
317 #\[1\] Max Gobel, Tsvit Hassen, Koenraad Oors, and Giorgio Orsi. Icdl 2011 table competition. In 2011 I
318 #cdl Christian Claeysen, Apostolos Antonopoulos, and Stefan Fleischmann. Research presentation at
319 #\\[2\\] Antonio Cesar, Jean-Luc Mounier, Liangcai Gao, Yulin Huo, Yu Fang, Florian Fleiner, and Ruiwei
320 #\\\[3\\\] Antonio Jijon, Peter Zeng, and Douglas S. Hersh. Scientific literature in the International Co
321 #\\\\[4\\\\] Logan Markewich, Hao Yuan, Xin Bing, Sheng, and Jian Li. Data-sharpening for Ime
322 #\\\\\[5\\\\\] Xu Jian, Zhang Bing, and Jian Li. Data-Sharpening for Image Annotation and Document Analysis. In 2011
323 #\\\\\\[6\\\\\\] Ross B. Girshick, Andrew Donahue, Trevor Darrell, and J. Bhattacharya. Technical Journal. Rich feature hierarc
324 #\\\\\\\[7\\\\\\\] Ross B. Girshick, Jason峨峨, and J. Bhattacharya. Technical Journal. International Conference on Com
325 #\\\\\\\\[8\\\\\\\\] Shiqiang Ren, Ross B. Girshick, and J. Bhattacharya. Technical Journal. Fast-recon-towards-the-world
326 #\\\\\\\\\[9\\\\\\\\\] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross B. Girshick. Technical Journal. IEEE Internati
327 #\\\\\\\\\\[10\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
328 #\\\\\\\\\\\[11\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
329 #\\\\\\\\\\\\[12\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
330 #\\\\\\\\\\\\\[13\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
331 #\\\\\\\\\\\\\\[14\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
332 #\\\\\\\\\\\\\\\[15\\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
333 #\\\\\\\\\\\\\\\\[16\\\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
334 #\\\\\\\\\\\\\\\\\[17\\\\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
335 #\\\\\\\\\\\\\\\\\\[18\\\\\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
336 #\\\\\\\\\\\\\\\\\\\[19\\\\\\\\\\\\\\\\\\\] Glenn Jocher, Alex Stokes, Ayush Chaurasia, Jirka Bozecov, NanoCode012, Taekie, Yonghye Kwon, K
337
338 #Biaocong, Mai Thanh Minh, Marc, albinkev, fatih, oleg, and wanghai yang. Ultralytics/yolov5: v6.0 - Yo
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354

```

Die Unterschiede sind z.T. ganze Tabellen.

90 >We did not control the document selection with regard to language. The vast majority of documents cont
91
92 >To ensure that future benchmarks in the document-layout analysis can be easily compared, we split up D
93
94 #For (2) i.e.g. AAPL from <https://www.bloomberg.com/>
95
96 #Table 1 shows the overall frequency of documents among the different sets. We ensure that subsets are
97
98 #In order to accommodate the different types of models currently in use by the community, we provide Doc
99
100
101 -Despite being cost-inducing and far less scalable than other languages, human annotation has several b
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
140
141
142
143
144
145
146
147

The annotation campaign was carried out in four phases. In phase one, we identified and prepared the d

the textual content of an element, which goes beyond visual layout recognition, in particular outside
At first sight, the task of visual document-layout interpretation appears intuitive enough to obtain p
Obviously, this is an issue with plausible annotations. For example, examples of plausible but inconsi
- Every list-item is an individual object with class `Label List-item`. This definition is differen
- A List-item is a paragraph with hanging indentation. Single lines can be manipulated as List-
- For every Caption , there must be exactly one corresponding Picture or Table.
- (4) Connected sub-pictures are grouped together in one Picture object.

problematische 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 eml
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,
```

erfolgreiche Parameter:

```
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": "grainite3.2-vision:2b",
92     # "vlm_pipeline_model_api": vlm_pipeline_model_api,
93
94 }
95 # "target": "zip",
```

Challenge: Die 16GB VRAM waren nicht genug, um alle features von docling zu unterstützen. Das verursachte periodische Endless-loop's in Docling serve.

Massnahme 1: Der Verzicht auf die Container-Version "Docling serve" und die Verwendung direkt in Python.

Massnahme 2: Die Ausführung von Docling auf der CPU, um das VRAM-Limit zu umgehen

Challenge: Die clouddcode_cli.exe in der VSCode-Umgebung hat durch einen extremen RAM-Verbrauch im Hintergrund die Ausführung von docling verhindert. freeze, not started, ...

<https://forum.cursor.com/t/high-memory-consumption-on-clouddcode-cli/106122>

Massnahme 1: Ein Uninstall von clouddcode_cli.exe war unumgänglich.

Challenge: Das parsen von Formeln in Docling mit CPU oder GPU ist sehr langsam. Den Verzicht auf die Extraktion der Formeln war keine Option, da eine maximale Qualität des Extrakts abgestrebt wurde, um die over-all Performance nicht zu beeinträchtigen.

Docling Log Ausschnitt:

```
[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
```

Massnahme 1: Einen zweiten Rechner 100% dafür einsetzen.

4. Methodik und Architektur

4.1 Graph-Konstruktion

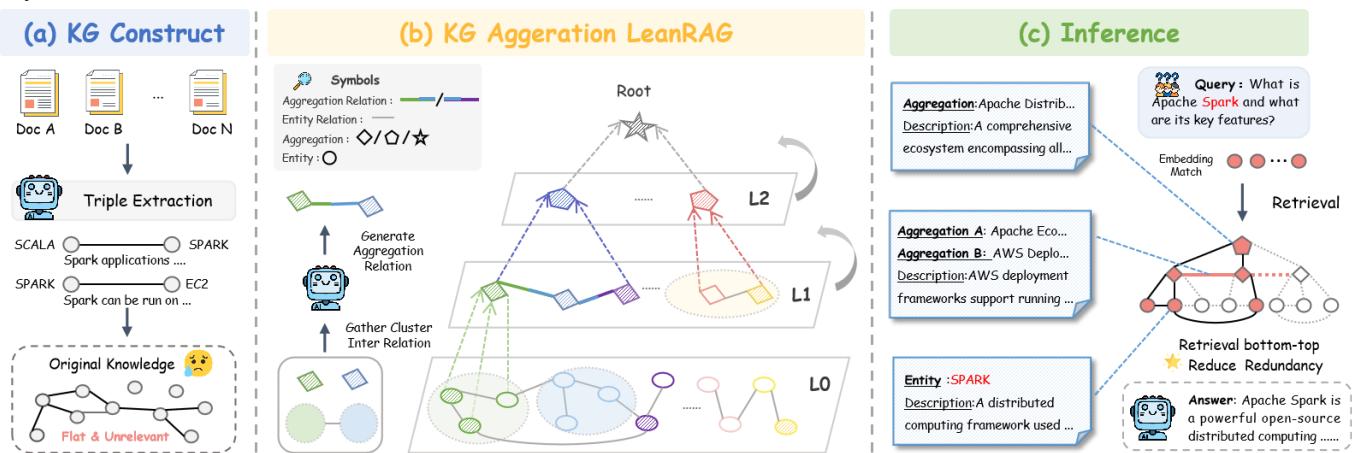
4.1.1 LeanRAG Ansatz

Detaillierung der Triple-Extraktion und der hierarchischen Retrieval-Struktur.

☆☆ Features

- **Semantic Aggregation:** Clusters entities into semantically coherent summaries and constructs explicit relations to form a navigable aggregation-level knowledge network.
- **Hierarchical, Structure-Guided Retrieval:** Initiates retrieval from fine-grained entities and traverses up the knowledge graph to gather rich, highly relevant evidence efficiently.
- **Reduced Redundancy:** Optimizes retrieval paths to significantly reduce redundant information—LeanRAG achieves ~46% lower retrieval redundancy compared to flat retrieval baselines (based on benchmark evaluations).
- **Benchmark Performance:** Demonstrates superior performance across multiple QA benchmarks with improved response quality and retrieval efficiency.

🏛️ Architecture Overview



LeanRAG's processing pipeline follows these core stages:

1. Semantic Aggregation

- Group low-level entities into clusters; generate summary nodes and build adjacency relations among them for efficient navigation.

2. Knowledge Graph Construction

- Construct a multi-layer graph where nodes represent entities and aggregated summaries, with explicit inter-node relations for graph-based traversal.

3. Query Processing & Hierarchical Retrieval

- Anchor queries at the most relevant detailed entities ("bottom-up"), then traverse upward through the semantic aggregation graph to collect evidence spans.

4. Redundancy-Aware Synthesis

- Streamline retrieval paths and avoid overlapping content, ensuring concise evidence aggregation before generating responses.

5. Generation

- Use retrieved, well-structured evidence as input to an LLM to produce coherent, accurate, and contextually grounded answers.
- Extraktion:** Umwandlung von Text in Entitäten und Relationen.

leanRAG Workflow

```
file_chunk.py
```

1. chunk raw input token-based with 512 Tokens and 64 Tokens overlap

Method 1: CommonKG

```
CommonKG/create_kg.py
```

2. create a list of match words (entities) for each chunk
3. create a list of "all entities" based on the match words without duplicates
4. "new triples" have "subject, predicate, object" triples init with corresponding reference to the chunk of origin
5. "next layer entities"
6. "new triples descriptions"

CommonKG/deal_triple.py

7. summarize descriptions => relation.jsonl

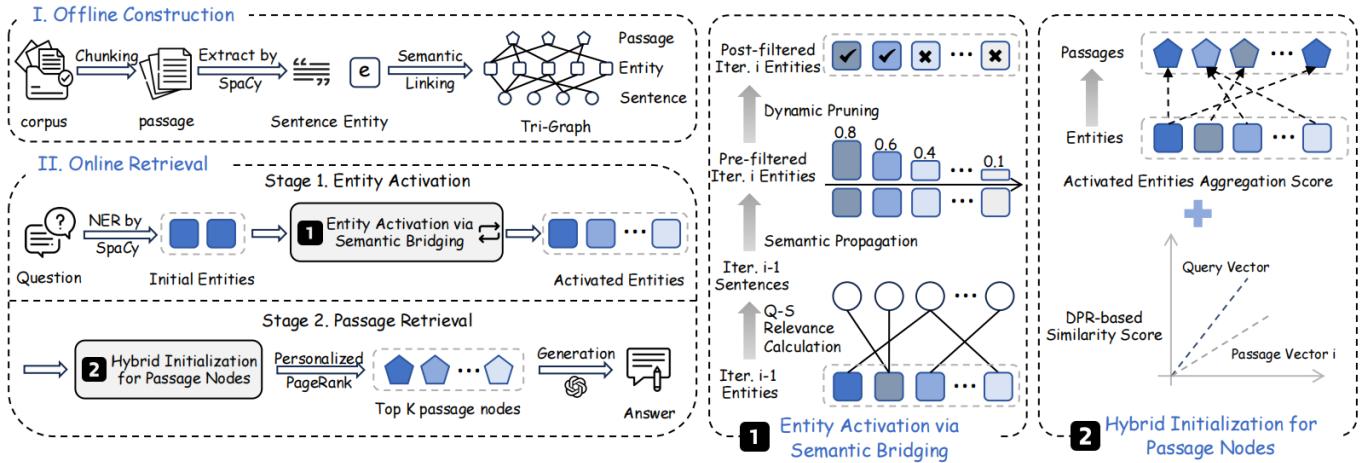
Method 2: GraphRAG

GraphExtraction/chunk.py

2. loads the chunks
 3. performs a "triple extraction" => entity.jsonl, relation.jsonl
- GraphExtraction/deal_triple.py
4. deal with duplicates of entries and relations
- build_graph.py
5. generating embeddings
 6. clustering labels (based on the embeddings)
 7. layer 1 clustering
 8. layer 2 clustering
 9. building vector DB

4.1.2 LinearRAG

LinearRAG: Linear Graph Retrieval-Augmented Generation on Large-scale Corpora - A relation-free graph construction method for efficient GraphRAG.



- ✓ Context-Preserving: Relation-free graph construction, relying on lightweight entity recognition and semantic linking to achieve comprehensive contextual comprehension.
- ✓ Complex Reasoning: Enables deep retrieval via semantic bridging, achieving multi-hop reasoning in a single retrieval pass without requiring explicit relational graphs.
- ✓ High Scalability: Zero LLM token consumption, faster processing speed, and linear time/space complexity.

Graphbuilding:

1. => load data
2. chunking data
3. get named entities - SpacyNER (Named Entity Recognition)
4. sentence splitting
5. get passages
6. get embeddings(sentences, entities, passages)
7. build graph => LinearRAG.graphml => ner_results.json => passage_embedding.parquet => sentence_embedding.parquet => entity_embedding.parquet

Retreival:

1. retrieval_results = qa(question)

linearRAG Results

LinearRAG, Dataset: 2wikimultihop, Results with local GPT-OSS-20b Model

```
[passage] Loaded 658 records from ./import\2wikimultihop\passage_embedding.parquet [entity] Loaded 40320 records from ./import\2wikimultihop\entity_embedding.parquet [sentence] Loaded 21206 records from ./import\2wikimultihop\sentence_embedding.parquet
```

2025-12-09 12:16:23,189 - INFO - Evaluation Results: 2025-12-09 12:16:23,191 - INFO - LLM Accuracy: 0.7350 (735.0/1000) 2025-12-09 12:16:23,191 - INFO - Contain Accuracy: 0.7210 (721/1000)

LinearRAG, Dataset: 2wikimultihop, Results with online gpt-4o-mini Model

```
[passage] Loaded 658 records from ./import\2wikimultihop\passage_embedding.parquet [entity] Loaded 40320 records from ./import\2wikimultihop\entity_embedding.parquet [sentence] Loaded 21206 records from ./import\2wikimultihop\sentence_embedding.parquet Retrieving: 100%
```

| 1000/1000 [02:43<00:00, 6.12it/s] QA Reading (Parallel): 100%|

1000/1000 [03:48<00:00, 4.37it/s] Evaluating samples: 100%

LLM_Acc=0.639, Contain_Acc=0.693] 2025-12-09 13:34:30,325 - INFO - Evaluation Results: 2025-12-09 13:34:30,325 - INFO - LLM Accuracy: 0.6390 (639.0/1000) 2025-12-09 13:34:30,325 - INFO - Contain Accuracy: 0.6930 (693/1000)

LinearRAG, Dataset: 2wikimultihop, Results with remote gemma3:17b Model

[passage] Loaded 658 records from ./import\2wikimultihop\passage_embedding.parquet [entity] Loaded 40320 records from ./import\2wikimultihop\entity_embedding.parquet [sentence] Loaded 21206 records from ./import\2wikimultihop\sentence_embedding.parquet Retrieving: 100%

1000/1000 [03:10<00:00, 5.24it/s] QA Reading (Parallel): 100%

1000/1000 [1:22:15<00:00, 4.94s/it] Evaluating samples: 100%

1000/1000 [03:24<00:00, 4.88sample/s, LLM_Acc=0.240, Contain_Acc=0.351] 2025-12-09 19:02:34,979 - INFO - Evaluation Results: 2025-12-09 19:02:34,980 - INFO - LLM Accuracy: 0.2400 (240.0/1000) 2025-12-09 19:02:34,981 - INFO - Contain Accuracy: 0.3510 (351/1000)

LinearRAG, Dataset: 2wikimultihop, Results with online gpt-4o Model

[passage] Loaded 658 records from ./import\2wikimultihop\passage_embedding.parquet [entity] Loaded 40320 records from ./import\2wikimultihop\entity_embedding.parquet [sentence] Loaded 21206 records from ./import\2wikimultihop\sentence_embedding.parquet Retrieving: 100%

1000/1000 [03:00<00:00, 5.55it/s] QA Reading (Parallel): 100%

1000/1000 [03:29<00:00, 4.78it/s] Evaluating samples: 100%

1000/1000 [00:40<00:00, 24.96sample/s, LLM_Acc=0.590, Contain_Acc=0.755] 2025-12-09 19:32:14,264 - INFO - Evaluation Results: 2025-12-09 19:32:14,264 - INFO - LLM Accuracy: 0.5900 (590.0/1000) 2025-12-09 19:32:14,265 - INFO - Contain Accuracy: 0.7550 (755/1000)

LinearRAG, Dataset: hotpotqa, Results with local GPT-OSS-20b Model

[passage] Loaded 1311 records from ./import\hotpotqa\passage_embedding.parquet [entity] Loaded 66846 records from ./import\hotpotqa\entity_embedding.parquet [sentence] Loaded 38455 records from ./import\hotpotqa\sentence_embedding.parquet Retrieving: 100%

1000/1000 [03:46<00:00, 4.42it/s] QA Reading (Parallel): 100%

1000/1000 [1:51:26<00:00, 6.69s/it] Evaluating samples:

1000/1000 [24:59<00:00, 1.50s/sample, LLM_Acc=0.771, Contain_Acc=0.662] 2025-12-10 20:59:41,463 - INFO - Evaluation Results: 2025-12-10 20:59:41,463 - INFO - LLM Accuracy: 0.7710 (771.0/1000) 2025-12-10 20:59:41,463 - INFO - Contain Accuracy: 0.6620 (662/1000)

LinearRAG, Dataset: musique, Results with local GPT-OSS-20b Model

[passage] Loaded 1354 records from ./import\musique\passage_embedding.parquet [entity] Loaded 67532 records from ./import\musique\entity_embedding.parquet [sentence] Loaded 39110 records from ./import\musique\sentence_embedding.parquet Retrieving: 100%



| 1000/1000 [03:15<00:00, 5.13it/s] QA Reading (Parallel): 100%



| 1000/1000 [3:51:21<00:00, 13.88s/it] Evaluating samples: 100%



| 1000/1000 [17:39<00:00, 1.06s/sample, LLM_Acc=0.642, Contain_Acc=0.317] 2025-12-11 02:00:28,341 - INFO - Evaluation Results: 2025-12-11 02:00:28,342 - INFO - LLM Accuracy: 0.6420 (642.0/1000) 2025-12-11 02:00:28,342 - INFO - Contain Accuracy: 0.3170 (317/1000)

LinearRAG, Dataset: medical, Results with local GPT-OSS-20b Model

[passage] Loaded 225 records from ./import\medical\passage_embedding.parquet [entity] Loaded 9033 records from ./import\medical\entity_embedding.parquet [sentence] Loaded 8985 records from ./import\medical\sentence_embedding.parquet Retrieving: 100%



| 2062/2062 [06:03<00:00, 5.67it/s] QA Reading (Parallel): 100%



| 2062/2062 [10:51<00:00, 3.17it/s] Evaluating samples: 100%



| 2062/2062 [01:26<00:00, 23.72sample/s, LLM_Acc=0.694, Contain_Acc=0.032] 2025-12-11 09:33:43,939 - INFO - Evaluation Results: 2025-12-11 09:33:43,939 - INFO - LLM Accuracy: 0.6940 (1431.0/2062) 2025-12-11 09:33:43,939 - INFO - Contain Accuracy: 0.0320 (66/2062)

4.1.3 GraphMERT

GraphMERT: Effiziente und skalierbare Gewinnung zuverlässiger Wissensgraphen aus unstrukturierten Daten

Ein einfaches Beispiel für eine Testimplementierung des Princeton GraphMERT-Papers.

<https://arxiv.org/abs/2510.09580>

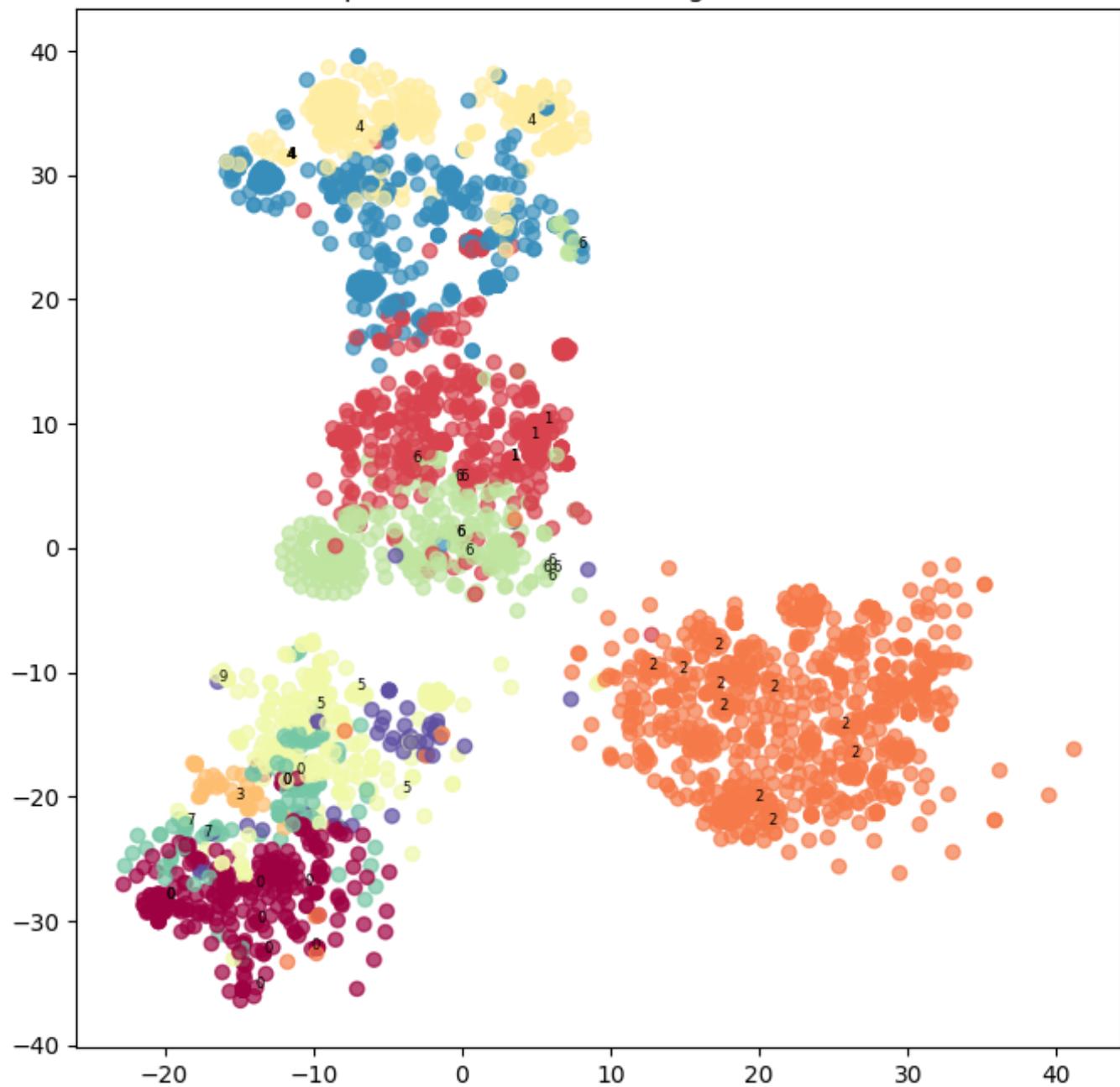
Seit fast drei Jahrzehnten erforschen Wissenschaftler Anwendungen neurosymbolischer künstlicher Intelligenz (KI), da symbolische Komponenten Abstraktion und neuronale Komponenten Generalisierung ermöglichen. Die Kombination beider Komponenten verspricht rasante Fortschritte in der KI. Dieses Potenzial konnte das Feld jedoch bisher nicht ausschöpfen, da die meisten neurosymbolischen KI-Frameworks nicht skalierbar sind. Zudem schränken die impliziten Repräsentationen und das approximative Schliessen neuronaler Ansätze Interpretierbarkeit und Vertrauen ein. Wissensgraphen (KGs), die als Goldstandard für die Repräsentation expliziten semantischen Wissens gelten, können die symbolische Seite abdecken. Die automatische Ableitung zuverlässiger KGs aus Textkorpora stellt jedoch weiterhin eine Herausforderung dar. Wir begegnen diesen Herausforderungen mit GraphMERT, einem kompakten, rein grafischen Encoder-Modell, das hochwertige KGs aus unstrukturierten Textkorpora und seinen eigenen internen Repräsentationen generiert.

GraphMERT und sein äquivalenter Wissensgraph bilden einen modularen neurosymbolischen Stack: neuronales Lernen von Abstraktionen; symbolische Wissensgraphen für verifizierbares Schliessen. GraphMERT + Wissensgraph ist das erste effiziente und skalierbare neurosymbolische Modell, das höchste Benchmark-Genauigkeit und überlegene symbolische Repräsentationen im Vergleich zu Basismodellen erzielt.

Konkret streben wir zuverlässige domänenspezifische Wissensgraphen (KGs) an, die sowohl (1) faktisch korrekt (mit Herkunftsnnachweis) als auch (2) valide (ontologiekonsistente Relationen mit domänenspezifischer Semantik) sind. Wenn ein grosses Sprachmodell (LLM), z. B. Qwen3-32B, domänenspezifische KGs generiert, weist es aufgrund seiner hohen Sensitivität, seiner geringen Domänenexpertise und fehlerhafter Relationen Defizite in der Zuverlässigkeit auf. Anhand von Texten aus PubMed-Artikeln zum Thema Diabetes erzielt unser GraphMERT-Modell mit 80 Millionen Parametern einen KG mit einem FActScore von 69,8 %; ein LLM-Basismodell mit 32 Milliarden Parametern erreicht hingegen nur einen FActScore von 40,2 %. Der GraphMERT-KG erzielt zudem einen höheren ValidityScore von 68,8 % gegenüber 43,0 % beim LLM-Basismodell.

GraphMERT Node Embeddings (t-SNE View)

GraphMERT Node Embeddings (t-SNE View)



GraphMERT Semantic Graph Visualization

GraphMERT Semantic Graph Visualization



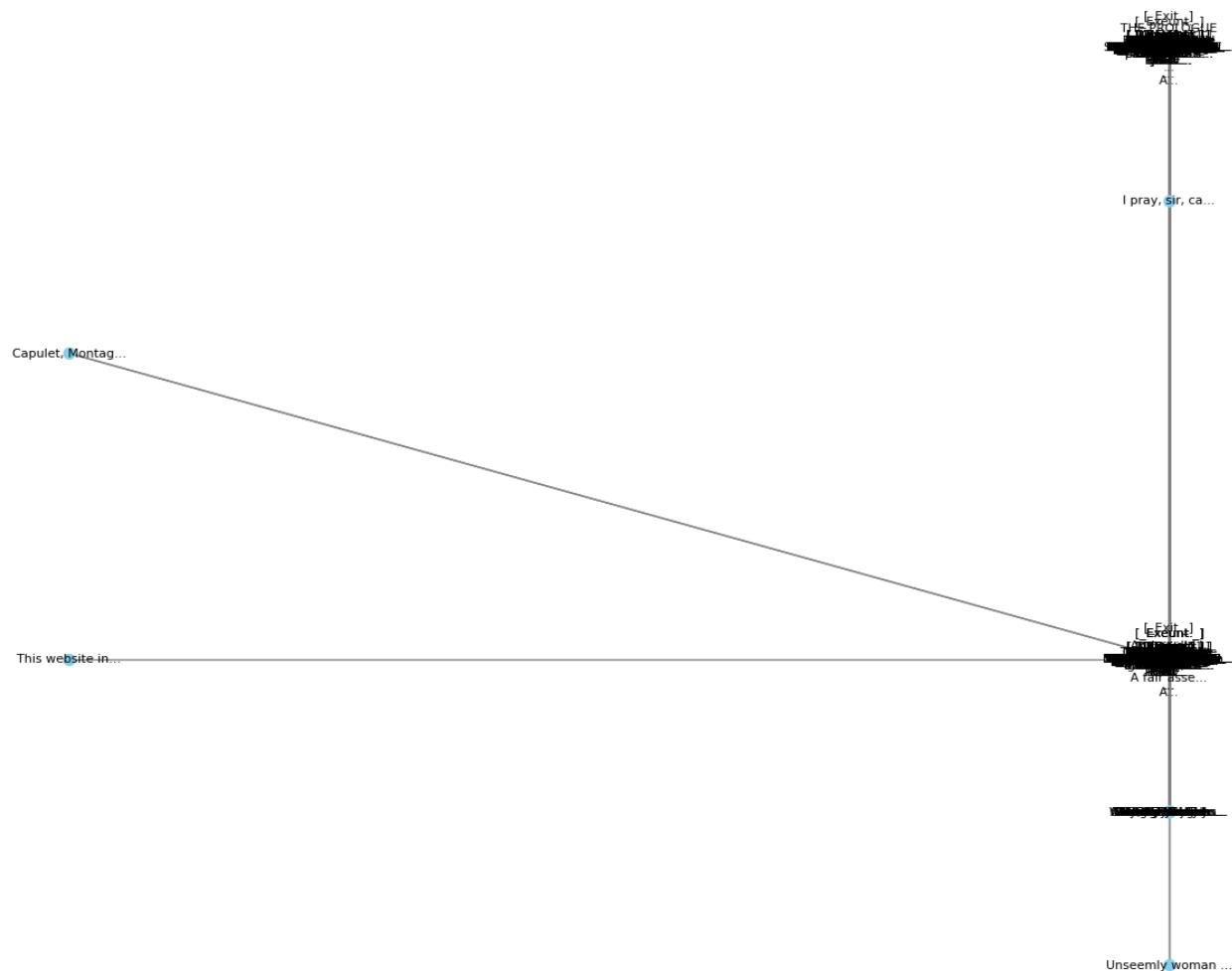
Query search on the graphs results Das ist es, was wir wollen, da die Suche im Graphen linear ist und auf verkettetem Wissen basiert, wobei die Knoten Daten über sich selbst enthalten.

Ein perfektes Resultat

Graph Visualization from GraphMERT Model Output Embeddings



Ein fast perfektes Resultat



- **Extraktion:** Umwandlung von Text in Entitäten und Relationen.
-

Aggregation: Semantische Aggregation zur Reduzierung von Redundanz.

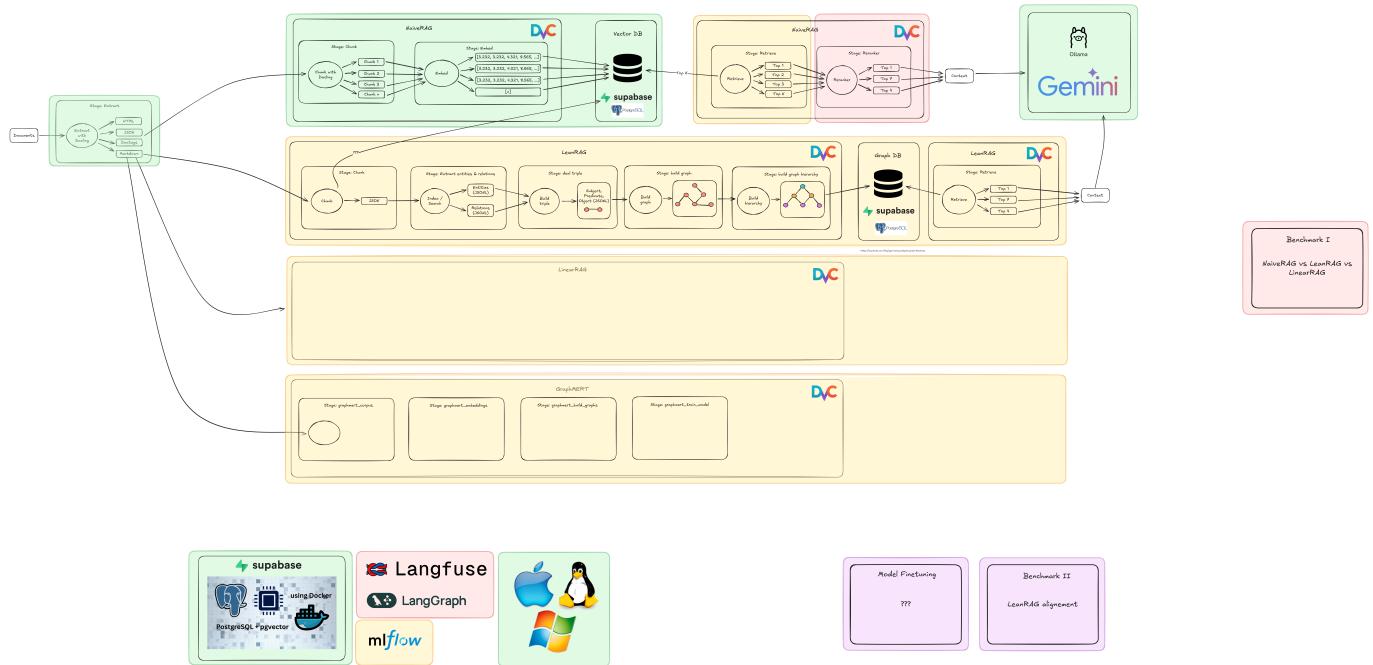
4.2 Fine-tuning Strategie

- Verwendung des **Unsloth Frameworks** für ressourceneffizientes Training.
- Integration von Ansätzen wie **GraphRAFT** oder **GraphMERT** zur Distillation von Wissen in kleine, domänenspezifische Modelle.

5. Implementierung

5.1 Systemarchitektur

Beschreibung der Pipeline von der PDF-Eingabe bis zur Antwortgenerierung.



5.2 Verwendete Hardware

Dokumentation der genutzten Ressourcen (z.B. 1x 4090 Desktop, M3 Pro 24GB) . 1 HP EliteBook X G11 => Massenextraktion mit Docling Prozessor Intel 5U

1 Lenovo Notbook Legion 9 16IRX8 Prozessor 13th Gen Intel(R) Core(TM) i9-13980HX (2.20 GHz)
Installierter RAM 32.0 GB (31.7 GB verwendbar) GPU Nvidia RTX4090 Mobile mit 16GB VRAM

5.3 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-Ansätze verzichtet diese Implementierung auf LLM-basierte Extraktion von Relationen und setzt stattdessen auf deterministische NLP-Prozesse und algorithmische Graphentwurf.

5.3.1 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.

5.3.2 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:

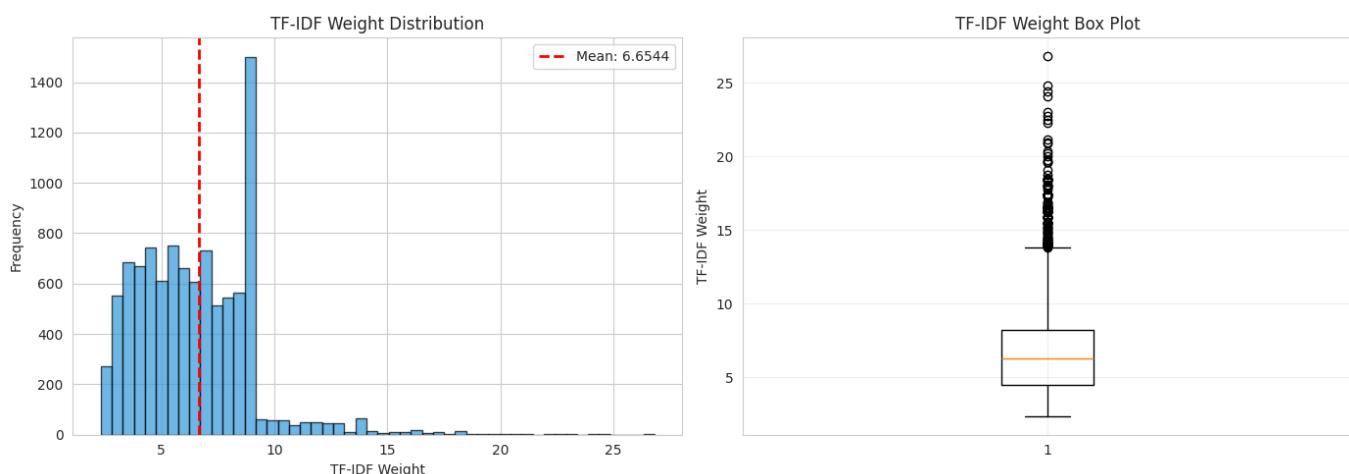
- **Passage \leftrightarrow Entity:** Diese Kanten sind gewichtet. Die Gewichtung erfolgt über eine **TF-IDF-Formel** ($\$log(1 + tf) * idf\$$), um die Relevanz einer Entität für einen spezifischen Abschnitt zu quantifizieren, anstatt nur binär das Vorhandensein zu speichern.
- **Structural Containment:**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



Diese Kennzahlen validieren, dass die algorithmische TF-IDF-Berechnung ($\log(1 + tf) \times \log(N / df)$) 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

Metrik	Wert	Beschreibung
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
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.

5.3.3 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.

5.3.4 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.

6. Evaluation und Benchmarking

6.1 Benchmark-Design

-

Ansatz 1: Generierung eines Testdatensatzes mittels Synthetic Data Generation (SDG) und Evaluierung durch ein "LLM als Judge".

-

Ansatz 2: Nutzung publizierter Benchmarks wie dem Open RAG Benchmark.

6.1.1 OpenRAGBench Linear RAG Graph

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

Metrik	Wert	Beschreibung
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.

6.2 Ergebnisse

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

7. Diskussion der Ergebnisse

- Qualität der generierten Graphen.
- Effektivität des Fine-tunings im Vergleich zu GPT-basierten Modellen.
- Ressourcenverbrauch und Skalierbarkeit.

8. Risikomanagement und Lessons Learned

Reflektion über die im Antrag identifizierten Risiken:

- Datenqualität und Graph-Dichte.
- Rechenintensität des Fine-tunings.
- Teamkoordination.
- 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.

9. Fazit und Ausblick

Zusammenfassung, ob ein 80M domänenpezifisches Modell tatsächlich grössere Modelle übertreffen konnte, und mögliche nächste Schritte.

Ausblick:

Aus den Ergebnissen konnten folgende Ansätze für die weitere Entwicklung abgeleitet werden:

- Die Qualität einer Knowledge Graphen wird hauptsächlich durch die Qualität der Entities beeinflusst. Ein Ansatz, um das Problem der sprachlichen Mehrdeutigkeit im Label der Entities ist, diese durch Attribute, abgeleitet aus dem Kontext, zu differenzieren. Ein Beispiel ist: "Der Müller hat dem Beruf eines Maurers" - Die Entity "Müller" ist folglich eine Maurer mit dem Familiennamen "Müller" und nicht eine Person mit dem Beruf Müller.
- Für eine produktive Lösung, sollten möglichst viele Verarbeitungsschritte im Scope eines einzelnen Dokuments (vor-)verarbeitet werden, bis und mit entity-relation Triples. Dies bringt folgende Vorteile mit sich:
 - Eine kontinuierliche Erweiterung des Graphen durch Vorverarbeitete Datensätze.
 - Parallelisierung
 - Die Möglichkeit, zu Entfernen, wenn Datensätze ungültig werden vereinfacht. Mögliche Gründe sind Fehler in den Daten oder Zeitbasierte Daten würde durch aktuellere ersetzt.
 - Mehrere Graphen können mit minimalem Offset für verschiedene Berechtigungsstufen erzeugt werden.
- Der Einfluss von Raum und Zeit muss systematisch im Graph-Modell berücksichtigt werden. z.B. Schwierigkeiten mit der Atmung werden auf Meereshöhe anders interpretiert wie auf dem Everest. Aktienkurse sind abhängig von der Zeit oder auch sich mit 100km/h zu bewegen war um 1900 rasend schnell und heute eher Durchschnitt.
- Für eine Knowledge Base mit verschiedenen Sprachen, können entities nur mit einer semantisch korrekten Übersetzung zusammengeführt werden. Um ständige Übersetzungen zwischen den Sprachen zu verhindern, könnte eine höhere Hierarchie mit einem Konzept-Graph repräsentiert werden. das heißt einzelne Fakten werden als Knowledge Graph dargestellt und darüber auf Konzepte abgebildet.
- Das Clustering identischer Relationen zu einem Hypergraph ist ein weiterer Ansatz, Teilgraphen zusammen zu führen, ohne sich die Möglichkeit zu verbauen Teile wieder zu entfernen. Ebenso können wahrscheinliche Relationen abgeleitet werden. (vom Hypergraph zurück zum Knowledge Graph)

10. Referenzen

- [1] Docling: An Efficient Open-Source Toolkit. <https://www.docling.ai/>
- [2] LeanRAG: Knowledge-Graph-Based Generation. <https://github.com/KnowledgeXLab/LeanRAG>
- [3] LinearRAG: A relation-free graph construction method for efficient GraphRAG. <https://github.com/DEEP-PolyU/LinearRAG>
- [4] GraphMERT: Efficient Distillation of Reliable KGs. <https://github.com/creativeautomaton/graphMERT-python>
- ... (Weitere Quellen gemäß Antrag).

11. Glossar

Hier ist ein Glossar, das spezifisch auf den Begriffen, Technologien und Konzepten basiert, die im vorliegenden Projektbericht (IMARA) verwendet werden. Es ist alphabetisch sortiert, um es direkt in Kapitel 11 einzufügen zu können.

11. 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. Im Projekt wurde es genutzt, um komplexe PDFs in maschinenlesbare Formate (Markdown/JSON) zu wandeln. Es traten Herausforderungen bezüglich VRAM-Verbrauch und Performance auf.
- **Embeddings:** Vektorrepräsentationen von Texten (Sätze, Entitäten, Passagen). Sie dienen als Basis für die Ähnlichkeitssuche und das Clustering in den Graphen.
- **FActScore:** Eine Metrik zur Bewertung der faktischen Korrektheit eines Wissensgraphen oder einer generierten Antwort. Im Bericht erzielt *GraphMERT* hierbei deutlich höhere Werte als reine LLMs.
- **Fine-tuning:** Das nachtrainieren eines LLMs (z. B. Qwen) auf spezifischen, graphenbasierten Daten, um die Antwortqualität und Domänenexpertise zu erhöhen.
- **GraphMERT:** Ein kompaktes, rein grafisches Encoder-Modell (Neurosymbolische KI), das effizient zuverlässige und ontologiekonsistente Wissensgraphen aus unstrukturierten Texten generiert.
- **GraphRAG (Graph Retrieval-Augmented Generation):** Eine Erweiterung von RAG, die statt flacher Textlisten strukturierte Wissensgraphen nutzt. Dies ermöglicht das Erkennen komplexer Beziehungen und *Multi-Hop-Reasoning*.
- **Graph-Sparsität:** Ein Maß für die Anzahl fehlender Kanten in einem Graphen relativ zur maximal möglichen Anzahl. Die Sparsität wird berechnet als: Sparsität = 1 - (Actual Edges / Possible Edges), wobei bei vollständiger Sparsität (100%) keine Kanten existieren und bei 0% alle möglichen Kanten vorhanden sind.

H - L

- **Hypergraph:** Eine im Ausblick erwähnte Graphenstruktur, bei der eine Kante (Edge) mehr als zwei Knoten verbinden kann. Dies wird als Ansatz vorgeschlagen, um identische Relationen zu clustern.
- **IMARA:** Der Name des Projekts. Es steht für die Entwicklung einer domänen spezifischen GraphRAG-Pipeline mit Model Fine-tuning.
- **Knowledge Graph (Wissensgraph):** Eine strukturierte Darstellung von Wissen in Form von Knoten (Entitäten) und Kanten (Beziehungen), die ein aktives, abfragefähiges Modell der Welt darstellt.
- **LeanRAG:** Ein GraphRAG-Ansatz, der auf semantische Aggregation und hierarchisches Retrieval setzt, um Redundanzen zu minimieren (ca. 46 % weniger Redundanz im Vergleich zu flachen

Baselines).

- **LinearRAG:** Eine effiziente GraphRAG-Methode, die "relation-free" arbeitet. Sie nutzt leichtgewichtige Entity Recognition und semantische Verlinkung für schnelle Verarbeitung mit linearer Komplexität.
- **LLM (Large Language Model):** Große Sprachmodelle, die als generative Komponente im RAG-Prozess dienen (z. B. GPT-4o, Qwen, Gemma).

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). Eine Schwäche von naivem RAG, aber eine Stärke von GraphRAG.
- **Naives RAG:** Bezeichnet im Bericht konventionelle, vektorbasierte RAG-Architekturen, die Wissen als unzusammenhängende Fakten (Chunks) behandeln und oft an kontextueller Fragmentierung leiden.
- **Neurosymbolische KI:** Kombination aus neuronalen Netzwerken (Generalisierung, Lernen) und symbolischer KI (Abstraktion, Logik, Graphen), wie sie im *GraphMERT*-Ansatz verwendet wird.
- **OpenRAGBench:** Ein Referenzdatensatz (Benchmark), der im Projekt genutzt wurde, um die Messbarkeit und Vergleichbarkeit der Ergebnisse sicherzustellen.

S - V

- **Semantic Aggregation:** Ein Feature von *LeanRAG*, bei dem Entitäten in semantisch kohärente Zusammenfassungen (Cluster) gruppiert werden, um die Navigation im Graphen zu verbessern.
- **Synthetic Data Generation (SDG):** Ein Ansatz zur Generierung von künstlichen Testdaten, um die Leistung des Systems zu evaluieren (z. B. mittels "LLM als Judge").
- **TF-IDF (Term Frequency-Inverse Document Frequency):** Ein statistisches Maß zur Bewertung der Wichtigkeit eines Terms in einem Dokument relativ zu einer Dokumentensammlung. Die Formel lautet: $\text{TF-IDF} = \log(1 + \text{tf}) \times \log(N / \text{df})$, 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 zur Gewichtung der *passage_entity*-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, das im Projekt für das ressourceneffiziente *Fine-tuning* der Modelle verwendet wurde.
- **ValidityScore:** Eine Metrik zur Bewertung der Gültigkeit von Relationen (Ontologie-Konsistenz) innerhalb eines Wissensgraphen.
- **Vektorsimilaritätssuche:** Das Suchverfahren klassischer RAG-Systeme, das Textabschnitte basierend auf mathematischer Ähnlichkeit (Vektornähe) findet, aber explizite Beziehungen oft ignoriert.

Tipps für die Ausarbeitung

- **Visualisierungen:** Nutzt die Grafiken aus eurem Zwischenbericht (*LeanRAG/Docling Architektur*), um die technischen Sektionen (Kapitel 3 & 4) zu füllen.
- **Code-Beispiele:** Fügt kurze Snippets eurer Automatisierungslösung oder der Unsloth-Konfiguration in Kapitel 5 ein.
- **Metriken:** In Kapitel 6 solltet ihr Tabellen mit Latenzzeiten und Genauigkeitswerten (Accuracy/F1) eurer Benchmarks zeigen.

Soll ich dir beim Ausformulieren eines spezifischen Kapitels (z.B. der Methodik oder der Evaluation) behilflich sein?