

AI POWERED QUANTUM EXPERT

Status Update - Reproduktion TutorGPT

Agenda

- 1 | Was haben wir bisher erarbeitet?
- 2 | Aufbau der Frameworks & Entscheidung
LangChain & LlamaIndex
- 3 | Aufbau unseres eigenen RAG-Systems
- 4 | Live-Demo
- 5 | Evaluation
- 6 | Ausblick

Was haben wir bisher erarbeitet?

Was haben wir bisher erarbeitet?



Aufbau der Frameworks & Entscheidung

Vergleich LlamaIndex vs. LangChain

Import

LlamaIndex

```
#pip install llama-index
#import important modules/packages/libs
import os
import os.path
from llama_index.core import (
    VectorStoreIndex,
    SimpleDirectoryReader,
    StorageContext,
    load_index_from_storage,
)
```

LangChain

```
#pip install -U langchain-community
#pip install pypdf
#pip install tiktoken
# alternativ kann cpu auch durch gpu ersetzt werden
#pip install faiss-cpu
import openai
import os
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEMBEDDINGS
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import
RecursiveCharacterTextSplitter
```

Vergleich LlamaIndex vs. LangChain

Storage / Keys

LlamaIndex

```
# set OpenAI API key
os.environ["OPENAI_API_KEY"] = "xxxx"
# check if storage already exists
PERSIST_DIR = "./storage"
#if not:
if not os.path.exists(PERSIST_DIR):
    # load the documents and create the index
    # load pdfs from directory: data with
simpledirectoryreader
    documents =
SimpleDirectoryReader("data").load_data()
```

LangChain

```
os.environ["OPENAI_API_KEY"] = "API_KEY"
loader = PyPDFLoader("load.pdf")
documents = loader.load()
text_splitter =
RecursiveCharacterTextSplitter(chunk_size=500,
chunk_overlap=50)
chunks = text_splitter.split_documents(documents)
```

Vergleich LlamaIndex vs. LangChain

Index / Storage

LlamaIndex

```
#creates index for documents
index = VectorStoreIndex.from_documents(documents,)
# store index for later (persistent=on the hard drive)
index.storage_context.persist(persist_dir=PERSIST_DIR)
else:
    # load existing index
    storage_context =
StorageContext.from_defaults(persist_dir=PERSIST_DIR)
    index = load_index_from_storage(storage_context)
```

LangChain

```
embeddings = OpenAIEmbeddings(model="text-embedding-ada-
002")
#sorgt dafür, dass die Vektoren durchsucht werden
faiss_index = FAISS.from_documents(chunks, embeddings)
#speichert die zuvor berechneten bzw. erstellten Vektoren
faiss_index.save_local("faiss_index")
vectorstore = FAISS.load_local("faiss_index", embeddings,
allow_dangerous_deserialization=True)
```

Vergleich LlamaIndex vs. LangChain

Querying / Question / LLM Response

LlamaIndex

```
#query the index
query_engine = index.as_query_engine()
#example questions
response = query_engine.query("Can you please sum up the
introduction to multicast?")
print(response)
```

LangChain

```
retriever = vectorstore.as_retriever()
query = "User Input hier eingeben"
retrieved_docs = retriever.get_relevant_documents(query)
retrieved_content = "\n".join([doc.page_content for doc in
retrieved_docs])
prompt = f"""
Hier sind einige relevante Informationen aus meiner
Wissensdatenbank:
{retrieved_content}
Nutze diese Informationen, um die folgende Frage zu
beantworten:
{query} """
response = openai.chat.completions.create(
model="gpt-4o-mini",
messages=[{"role": "system", "content": "Du bist ein
hilfreicher Assistent."},
 {"role": "user", "content": prompt}])
print(response.choices[0].message.content)
```



Welches Framework nutzen wir?

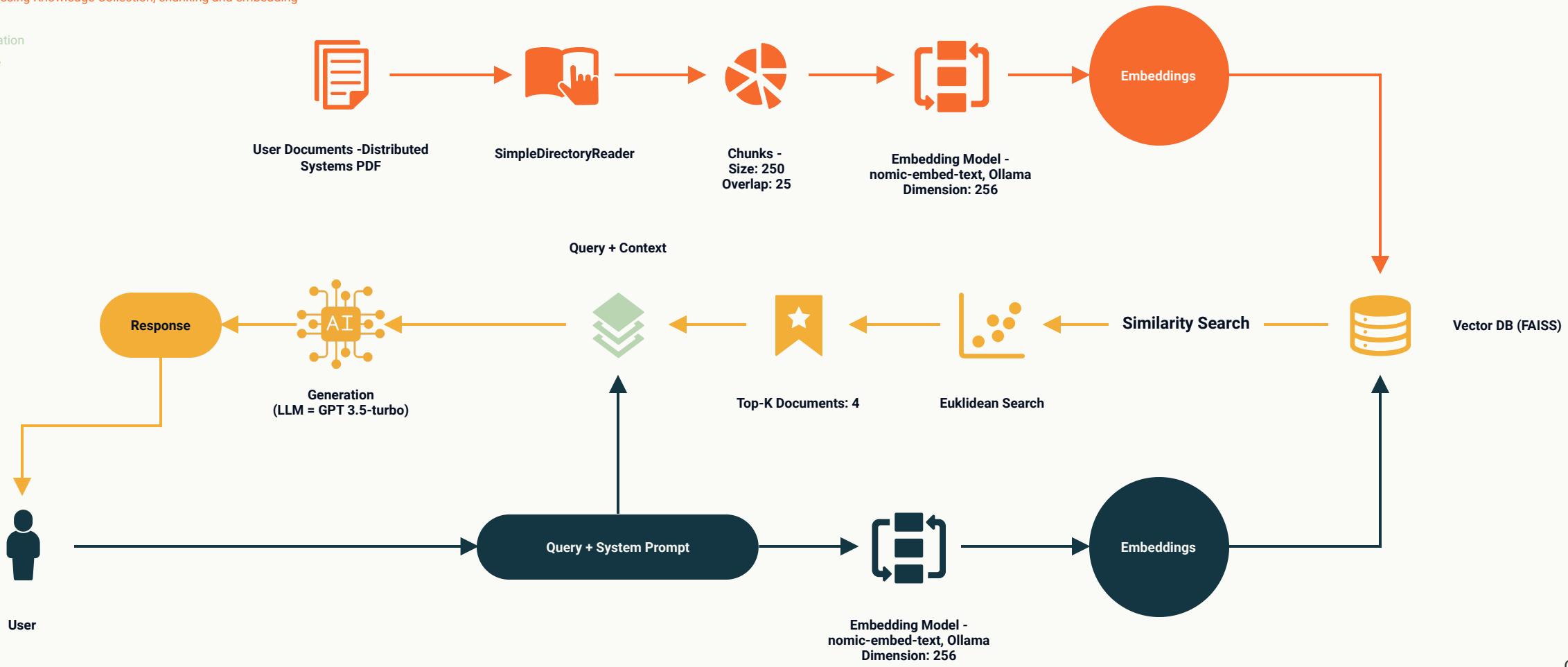
- Kriterien für die Entscheidung des zu nutzenden Frameworks
- Benutzerfreundlichkeit
 - Änderbar
 - Verständlich
- Dokumentation
- Dokumentenupload und -verarbeitung
- Ausgelegt spezifisch für den RAG-Ansatz?
- Leistungsfähigkeit und Skalierung (in Bezug auf große Datenmengen)

- **LlamaIndex**

Aufbau unseres eigenen RAG-Systems

Aufbau RAG-System

Processes:
Pre-Processing Knowledge Collection, chunking and embedding
Retrieval
Augmentation
Response



Live-Demo

Evaluation

Evaluierungsmethoden

Ziel der Evaluierung: Vergleich der aktuellen Ergebnisse mit denen des TutorGPT, um die Fortschritte und Verbesserungen zu bewerten.

Methode	Ansatz	Auswahl
LLM-derived Metrics	Verwendet vortrainierte Sprachmodelle (LLMs), um automatisch die Qualität von generierten Texten zu bewerten.	✗
Prompting LLMs:	In der prompt basierten Evaluierung werden gezielte Eingabeprompts verwendet, um die Textqualität zu bewerten.	✓
Fine-tuning LLMs	Ist der Prozess, bei dem ein vortrainiertes Modell durch Aktualisierung der Parameter mit neuen, aufgabenspezifischen Daten an spezifische Aufgaben angepasst wird.	✗
Human-LLM Collaborative Evaluation	Eine Methode, bei der Menschen und LLMs gemeinsam an der Evaluierung arbeiten	✗

Prompt basierte Evaluierung - LIKERT SCALE



Allgemein

Ist eine Bewertungsskala, die verwendet wird, um Meinungen, Einstellungen oder Verhaltensweisen zu messen. Sie besteht aus einer Aussage oder Frage, gefolgt von einer Reihe von Antwortmöglichkeiten, die typischerweise fünf oder sieben Stufen umfassen.



Ausführung

- Frage oder Aussage formulieren
- Eingabeprompts Festlegen und ausführen
- Daten sammeln•Daten Analysieren



Auswahl

- Vergleichbarkeit der Ergebnisse mit TutorGPT
- Bietet eine fundierte Grundlage für die Analyse

Prompt basierte Evaluierung - PAIRWISE



Allgemein

Mit der Pairwise-Methode werden zwei Antworten direkt miteinander verglichen, um die qualitativ bessere zu ermitteln. Diese Methode wird insbesondere eingesetzt, um unterschiedliche Systeme oder Modelle miteinander zu vergleichen.



Ausführung

- Zwei Antwortoptionen generieren
- Eingabeprompts Festlegen und ausführen
- Daten Analysieren



Auswahl

- Durchführung der Tests mit einer weiteren Evaluierungsmethode
- Überprüfung der Likert Scale Ergebnisse

Use Case 1: Distributed Systems

Voraussetzungen, um den Vergleich der beiden RAG Systeme TutorGPT und TutorGPT 2.0 zu gewährleisten

Voraussetzungen	TutorGPT*	TutorGPT 2.0 Test 1	TutorGPT 2.0 Test 2
Context: Distributed Systems Slides	X	X	X
Question: Same 7 Questions	X	X	X
Evaluation method: Likert Scale (Ranking 1-5)	X	X	X
Vector Index: Faiss	X	X	X
Chunk Size: 250 with 25 Overlap	X	X	
Chunk Size: 500 with 50 Overlap			X
Embedding: nomic-embed-text (gpt-3.5)	X	X	X
Evaluation prompt	X	X	X

*Vergleichsbasis gemma: 7b/gold liefert zu allen Fragen Antworten und erzielt das beste Ergebnis / Empfehlung von TutorGPT mind. 7b Parameter zu verwenden

Ergebnisanalyse - Likert Scale

	Tutor GPT gemma:7b; Gold	Tutor GPT 2.0 - Test 1 250/25	Tutor GPT 2.0 - Test 2 500/50
Faithfulness Score	4,1	4,3	4,5
Answer Relevance Score	5,0	4,6	4,7
Conversational Quality Score	4,6	4,3	4,4
Explanatory Depth Score	4,1	4,0	4,2
Total Score	4,5	4,3	4,5

- **Gesamtbewertung:** Tutor GPT 2.0 - Test 2 erreicht denselben Total Score wie Tutor GPT gemma:7b; Gold
- Tutor GPT gemma:7b; Gold: Führt in den Kategorien Antwortrelevanz und Konversationsqualität
- Tutor GPT 2.0 - Test 1: Verzeichnet die niedrigste Gesamtbewertung, insbesondere in Konversationsqualität und Erklärungstiefe
- Tutor GPT 2.0 - Test 2: Erreicht einen Gesamtscore von 4,5 und zeigt starke Leistungen in nahezu allen Bewertungsbereichen

Use Case 2: MISQ – Digital Business

Hypothese 1: Eine Erhöhung der Chunk Size führt zu verbesserten Ergebnissen.

Hypothese 2: Pairwise und Likert Scale führen zu vergleichbaren Ergebnissen.

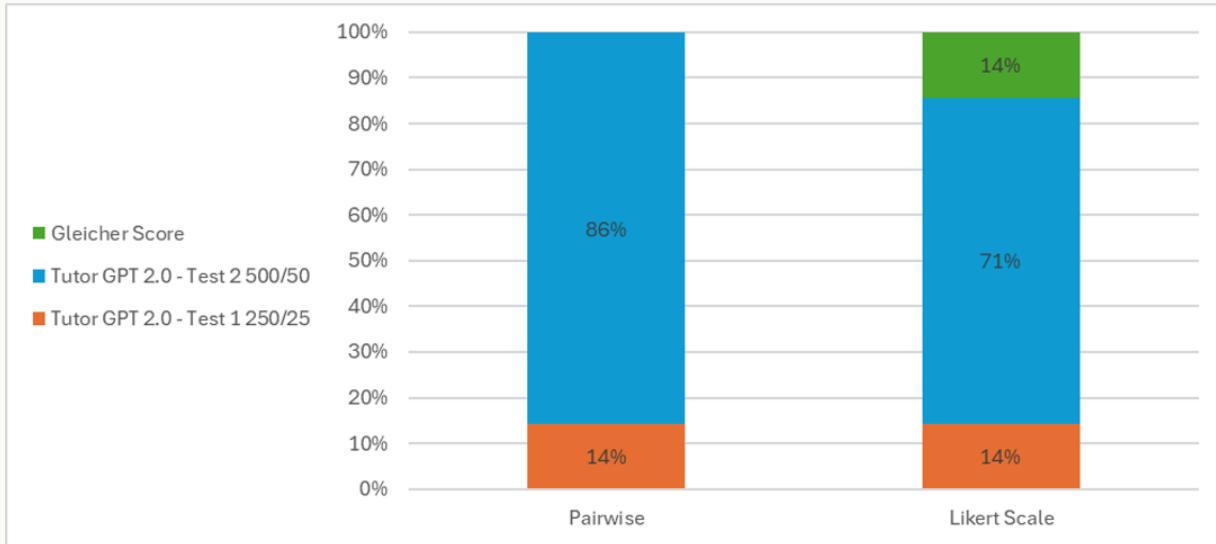
Voraussetzungen:	Use Case 1	Use Case 2
Context: MISQ-Digital Business Slides		X
Question: 7 new Questions		X
Evaluation method: Likert Scale (Ranking 1-5)	X	X
Evaluation method: Pairwise		X
Vector Index: Faiss	X	X
Chunk Size: Tutor GPT 2.0 - Test 1 250/25	X	X
Chunk Size: Tutor GPT 2.0 - Test 2 500/50	X	X
Embedding: nomic-embed-text (gpt-3.5)	X	X
Evaluation prompt	X	X

Ergebnisanalyse - Likert Scale

Use Case 2	Tutor GPT 2.0 - Test 1 250/25	Tutor GPT 2.0 - Test 2 500/50
Faithfulness Score	4,6	4,6
Answer Relevance Score	4,4	4,5
Conversational Quality Score	4,5	4,5
Explanatory Depth Score	3,6	4,1
Total Score	4,3	4,4

- **Gesamtbewertung:** Tutor GPT 2.0 - Test 2 erzielt eine höhere Gesamtbewertung.
- Tutor GPT 2.0 - Test 1: Gesprächsqualität und Faktengetreue sind vergleichbar bewertet
- Tutor GPT 2.0 - Test 2: Überzeugt mit besserer Antwortrelevanz und Erklärungstiefe
 - ✓ Hypothese 1: Eine Erhöhung der Chunk Size führt zu verbesserten Ergebnissen.

Ergebnisanalyse - Pairwise



- Pairwise-Analyse: In der Analyse wurden 86 % der Antworten von Tutor GPT 2.0 – Test 2 (500/50) als qualitativ besser und konsistenter bewertet.
 - ✓ Hypothese 1: Eine Erhöhung der Chunk Size führt zu verbesserten Ergebnissen.
- Überleitung zu Likert Scale: Trotz der unterschiedlichen Ansätze der beiden Methoden liefert die Analyse auf der Ebene der einzelnen Fragen ähnliche Ergebnisse.
 - ✓ Hypothese 2: Pairwise und Likert Scale führen zu vergleichbaren Ergebnissen.

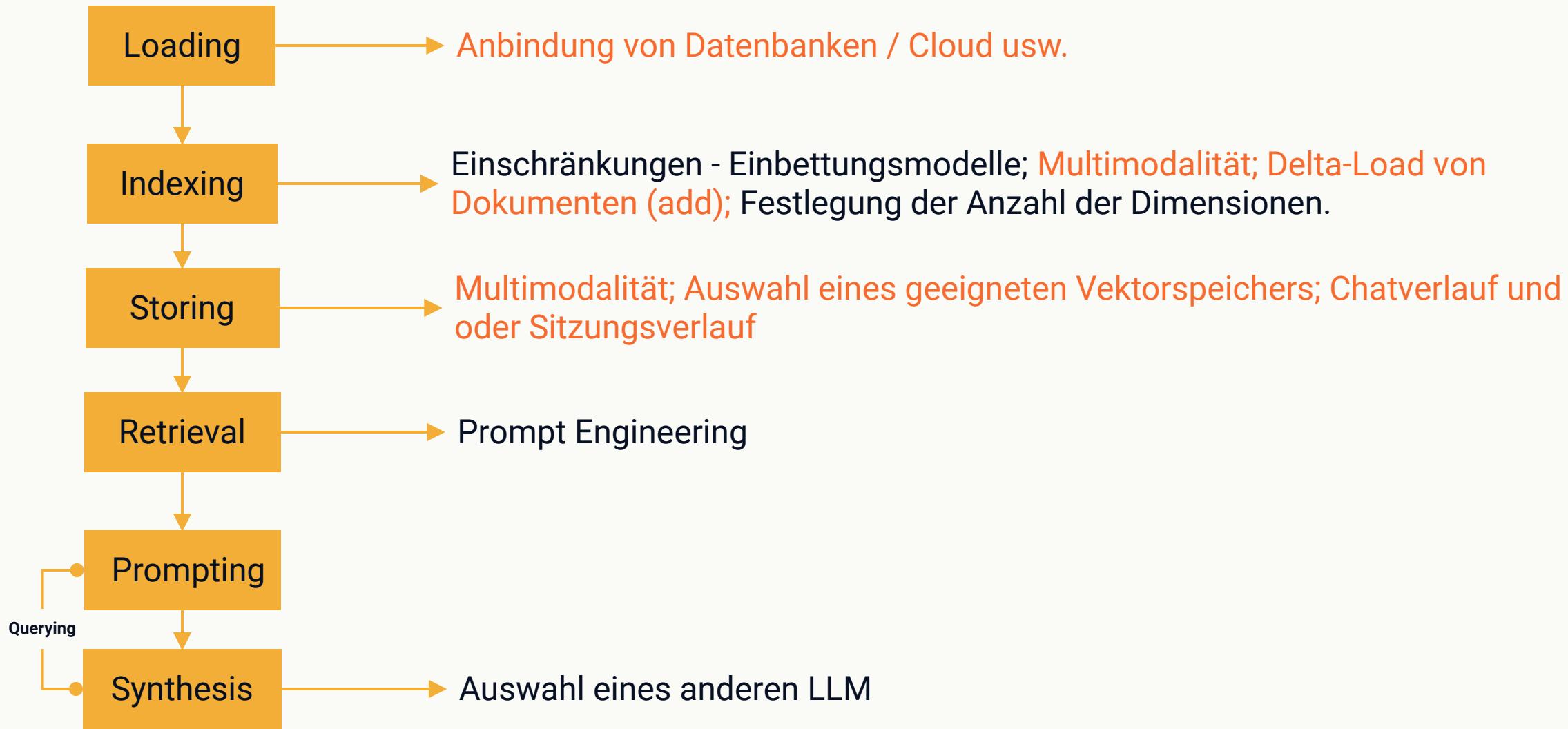
Ausblick Optimierungen

	Tutor GPT 2.0 - Test 1 250/25	Tutor GPT 2.0 - Test 2 500/50	Tutor GPT 2.0 - Test 3 neues Embedding
Faithfulness Score	4,6	4,6	4,7
Answer Relevance Score	4,4	4,5	4,8
Conversational Quality Score	4,5	4,5	4,7
Explanatory Depth Score	3,6	4,1	4,5
Total Score	4,3	4,4	4,7

- Optimierungen im Tutor GPT 2.0 - Test 3:
 - Alternatives Embedding-Modell
 - Erhöhung der Dimension auf 1536
- Bewertung: Tutor GPT 2.0 - Test 3 liefert insgesamt die besten Ergebnisse. Besonders die Antwortrelevanz, die Tiefe der Erklärungen und die Gesamtqualität profitieren von den Anpassungen. Steigerung des Total Scores im Vergleich zu Test 2 um etwa 7 %.

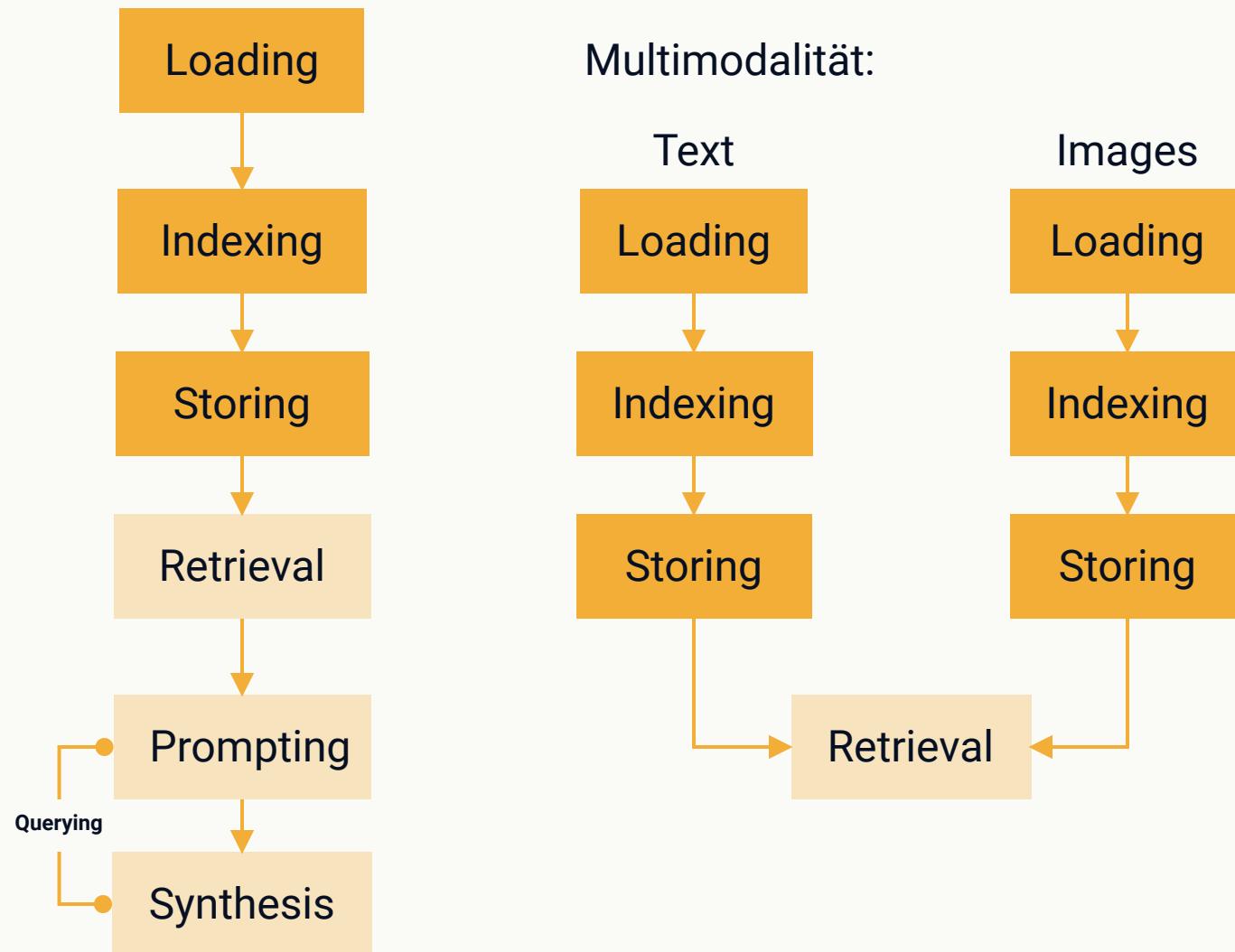
Ausblick

Insights / Outlook



Quelle: [3]

Outlook



Quelle: [8]

Speicheroptionen Für Daten in LlamaIndex

Speicheroption	Beschreibung	Vorteile	Einschränkungen	Eignung für unser Projekt
FAISS (lokale Festplatte)	Open-Source-Bibliothek für schnelle Vektorschre / Ähnlichkeitsabfragen	- Schnelle Vektorschre - Open Source und kostenlos - Hohe Performance bei großen Datenmengen	- Keine native Unterstützung für verteilte Systeme - Manuelles Setup erforderlich	Gut geeignet
Lokaler Speicher	Daten und Indexe auf Festplatte gespeichert (z. B. als JSON-Datei)	- Einfache Implementierung - Keine externen Dienste notwendig - Offline nutzbar	- Nicht ideal für große Datenmengen - Kein verteiltes System - Datenverwaltung manuell	Weniger geeignet
Cloud Speicher	Index-Dateien werden in Cloud-Speicherdiene hochgeladen (z.B. AWS S3, Google Cloud Storage, Azure Blob Storage)	- Skalierbar - Zentraler Zugriff von verschiedenen Geräten - Automatische Backups	- Abhängig von externen Diensten - Kosten können anfallen	Gut geeignet
Datenbanken (SQL)	Indexe und Metadaten werden in relationalen Datenbanken gespeichert (z.B. MySQL, PostgreSQL)	- Gut für strukturierte Daten - Einfache Integration mit bestehenden Systemen - Verlässlich	- Skalierung begrenzt - Relationale Datenbanken sind nicht für Vektorschre optimiert	Weniger geeignet
NoSQL-Datenbanken	Flexible Speicherung der Dokumente und Indexe (z.B. MongoDB, Firebase Flexible)	- Skalierbar und flexibel - Gute Unterstützung für unstrukturierte Daten	- Keine spezialisierte Unterstützung für Vektordaten - Komplexere Implementierung	Weniger geeignet
Vektordatenbanken	Spezialisierte Speicherlösungen zur Speicherung der Embeddings (z.B. Pinecone, Weaviate, Milvus, Chroma)	- Optimiert für Vektorschre - Schnell und skalierbar - Unterstützt semantische Abfragen	- Externe Dienste erforderlich - Kosten für Cloud-Lösungen - Einrichtung komplexer	Sehr gut
Integrierter Index-Speicher	LlamaIndex speichert Indexe direkt im RAM während der Laufzeit, ohne externes Speichermedium	- Ideal für schnelle Prototypen - Kein externer Speicher nötig	- Daten gehen verloren, wenn Programm endet - Nicht für große Datenmengen geeignet	Gut für Prototypen, aber keine dauerhafte Lösung

Empfehlung

- Vektordatenbanken (z. B. Pinecone, Chroma)
 - Das Projekt könnte eine effiziente semantische Suche, sowie ein schnelles Wachstum der Datenmengen erfordern
 - In diesem Kontext ermöglichen Vektordatenbanken eine hohe Skalierbarkeit und Performance - besonders für die Speicherung und Suche von Embeddings.
 - Sie sind perfekt geeignet, wenn Skalierung und flexible Datenverarbeitung essenziell sind.

- FAISS (lokal)
 - Wenn die Datenmenge begrenzt bleibt und das Projekt lokal betrieben werden kann, bietet FAISS eine kostenfreie, leistungsstarke Lösung.
 - Datenschutz bleibt unter Kontrolle, was bei sensiblen Daten ein großer Vorteil ist.
 - Es ist besonders nützlich, wenn keine Abhängigkeit von externen Diensten entstehen soll.

FAQ

Quellen

- 1 | <https://docs.llamaindex.ai/en/stable/>
- 2 | <https://python.langchain.com/docs/introduction/>
- 3 | <https://github.com/LEAN-96/RAG-Demystified/>
- 4 | <https://arxiv.org/html/2402.01383v2>
- 5 | chrome-extension://efaidnbmnnibpcajpcglclefindmkaj/<https://aclanthology.org/2024.emnlp-main.896.pdf>
- 6 | <https://www.pinecone.io/learn/series/rag/embedding-models-rundown/>
- 7 | <https://huggingface.co/spaces/mteb/leaderboard>
- 8 | <https://www.youtube.com/watch?v=35RlrrgYDyU>
- 9 | <https://docs.llamaindex.ai/en/v0.10.18/understanding/storing/storing.html>