

RAG Hackathon Coach - Sessionprotokoll

Thema: Retrieval-Augmented Generation - Woche 1-3

[User] was sind die key concepts von den hochgeladenen slides

[RAG] Zusammenfassung der Key Concepts:

Woche 1: Grundlagen von RAG

- Motivation: Limitierungen klassischer LMs
- Lösung: RAG = Retrieval + Generation
- Architektur: Embedding → Vektor DB → Prompt Injection → LLM
- Vorteile: weniger Halluzination, Zugriff auf aktuelle Daten, kein Fine-Tuning nötig
- Tools: LangChain, ChromaDB, Assistant API, direkte Schnittstellen
- Übung via GitHub

Woche 2: Tokenization & Embeddings

- Tokenization → Subword BPE
- One-hot-Encoding vs. Embedding
- Kontextuelle Embeddings (static vs. contextual)
- Cosine Similarity
- Positional Embeddings (Sin/Cos)

Woche 3: Verbesserungen des Retrievals

- Vektor Stores vs. klassische DBs
- ANN-Algorithmen (FAISS, HNSW, ANNOY...)
- Retrieval-Tuning: Hybrid Search, Thresholding, Query Expansion
- Query Transformation (HyDe, Multi-Step, Rewrite)
- Reranking mit Cross-Encoders
- End-to-End Evaluation

[User] stelle mir alles auf was in der woche 1 passiert

[RAG] Inhalte Woche 1 im Detail:

1. Motivation

- LMs veraltet, teuer zu tunen, halluzinieren, kein Zugriff auf interne Daten

2. Lösung: RAG

- Echtzeit-Retrieval
- keine Modelländerung nötig
- Prompt-Injection mit externen Texten

3. Was ist RAG?

- Einführung durch Meta 2020
- Architektur: Retrieval (Embeddings) + Generation (LLM)

4. RAG-Fähigkeiten

- Open-Book-System
- Zugriff auf API, Knowledge Graph, aktuelle DBs
- Wenig Trainingsdaten nötig

5. Tools & Techniken

- LangChain RetrievalQA Chain
- Chroma als Vektorstore
- Assistant API
- Limitierungen: keine Deduplikation, Versionsprobleme

6. Übung

- GitHub-Link zum eigenen Vektor-RAG-System

[User] gib mir noch die wesentlichen codeblöcke der woche an

[RAG] Woche 1 - Wichtige Codeblöcke:

PDF → ChromaDB

```
```python from langchain.documentloaders import PyPDFLoader from langchain.textsplitter import CharacterTextSplitter from langchain.vectorstores import Chroma from langchain.embeddings import HuggingFaceEmbeddings
```

```
loader = PyPDFLoader("deindokument.pdf") documents = loader.load() textsplitter = CharacterTextSplitter(chunksize=500, chunkoverlap=100) docs = textsplitter.splitdocuments(documents)
```

```
embedding = HuggingFaceEmbeddings(modelname="sentence-
transformers/all-MiniLM-L6-v2") db = Chroma.fromdocuments(docs,
embedding, persistdirectory="chromadb") db.persist() ``
```

### **Query → Antwort**

```
``python from langchain.chains import RetrievalQA from langchain.llms
import OpenAI

db = Chroma(persistdirectory="chromadb",
embeddingfunction=embedding) retriever = db.asretriever() llm =
OpenAI(model_name="gpt-4")

qachain = RetrievalQA.fromchaintype(llm=llm, retriever=retriever,
returnsource_documents=True)

query = "Was waren die ersten Worte nach der Mondlandung?" result =
qa_chain({"query": query}) print(result["result"]) ``
```

### **Prompt Template**

```
``python from langchain.prompts import PromptTemplate

template = ""Beantworte die folgende Frage ausschließlich basierend auf
dem gegebenen Kontext:

{context}

Frage: {question} Antwort:""

prompt = PromptTemplate(input_variables=["context", "question"],
template=template) ``

(... hier folgen alle Codeblöcke und Antworten aus Woche 2 und 3,
vollständig enthalten ...)
```