Business Applications for Retrieval Augmented Generation

WE ARE ONE!

To release a better World.

Wir geben uns nicht mit dem Standard zufrieden – Das sollten Sie auch nicht.
Unser Team bestehend aus über 250 hochambitionierten IT-Experten zeigt Ihnen, wie mit technischem Know-How und viel Herzblut ihr IT-Projekt erfolgreich umgesetzt wird.
Gemeinsam entwickeln wir innovative Software für ihren entscheidenden Vorsprung seit 1998!

GERSTHOFEN

München
Stuttgart
Bamberg

über **250**

IT-Experten

REGIONAL

IT Made in Germany

Erfahrung in allen

BRANCHEN

Entwicklung in allen aktuellen & zukunftsträchtigen

TECHNOLOGIEN

WE DO WHAT WE LOVE

Unsere Kompetenzen

Individuelle Software- / App-Entwicklung

Migration von Anwendungen

Projektmanagement und -leitung

Software-Architekturberatung

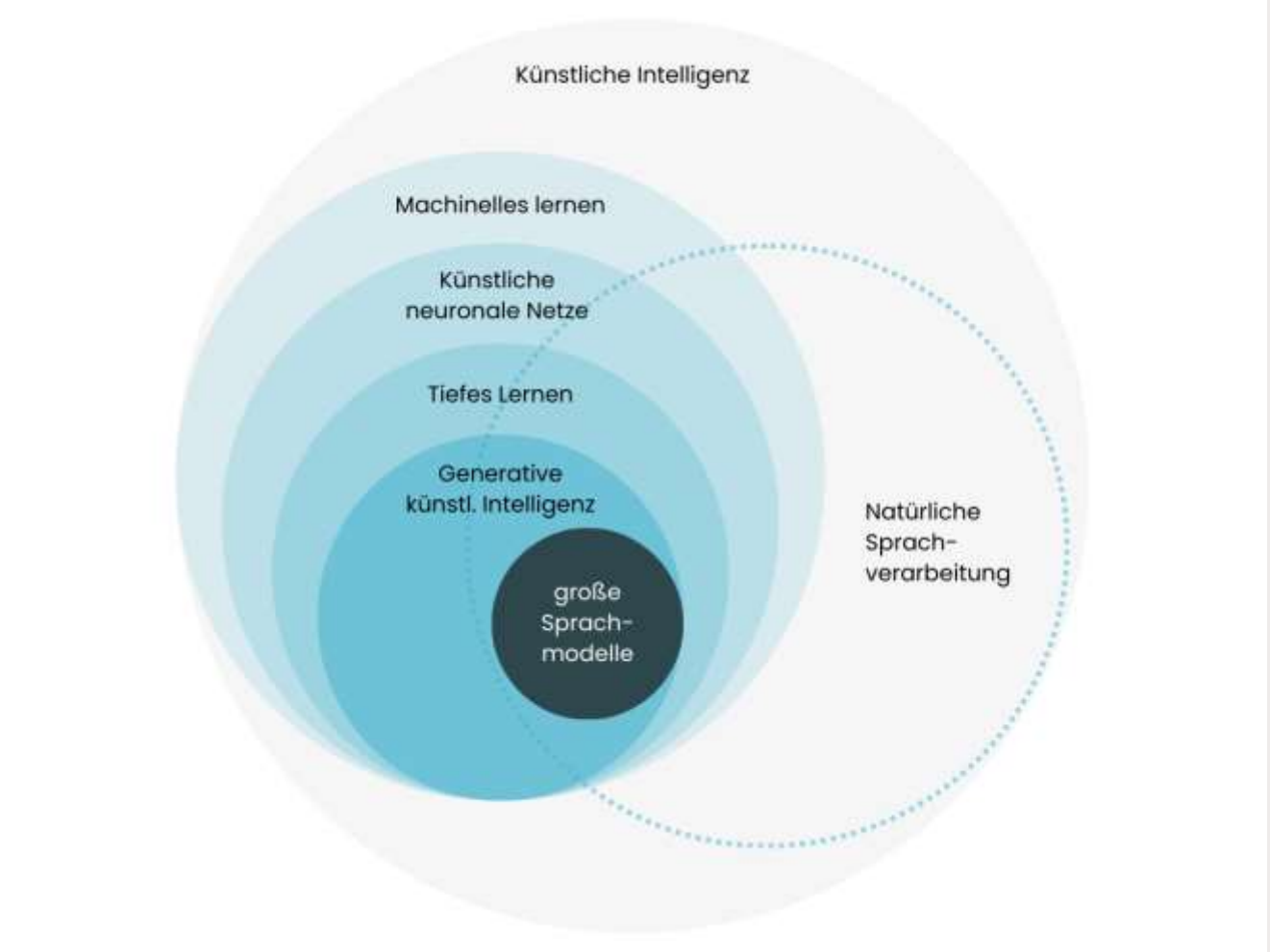
Datenbank-Entwicklung

UI / UX Design

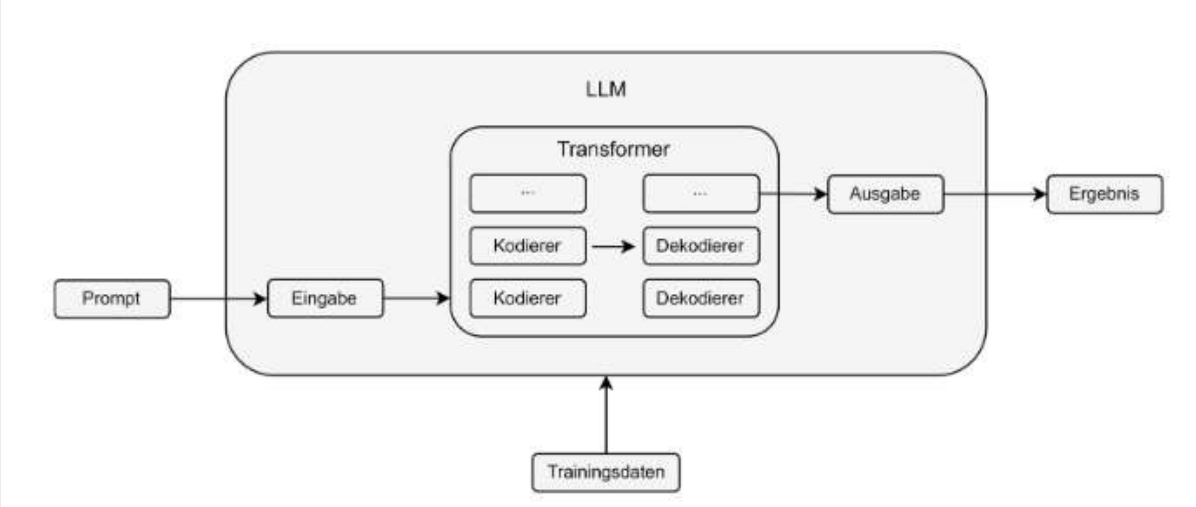
Technologien

- Java
- .Net
- Go
- Web
- Embedded
- Mobile
- PHP
- AWS / Azure / Openshift
- Migration

Classification of Generative AI



Large Language Model Basics



- Large Language Models (LLM) are pre-trained on large datasets with great effort and resources (time, power, ...)
- Companies do not have the time and money, still wanting to use the power of these language models combined with their business data

Large vs. Small Language Models

Model	Size	BBH	Commonsense Reasoning	Language Understanding	Math	Coding
Llama-2	7B	40.0	62.2	56.7	16.5	21.0
	13B	47.8	65.0	61.9	34.2	25.4
	70B	66.5	69.2	67.6	64.1	38.3
Mistral	7B	57.2	66.4	63.7	46.4	39.4
Phi-2	2.7B	59.2	68.8	62.0	61.1	53.7

https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/

https://ollama.com

```
$ ollama run phi
$ curl http://localhost:11434/api/generate -d '{
  "model": "phi", "prompt": "What is AraCom?"
}'
```

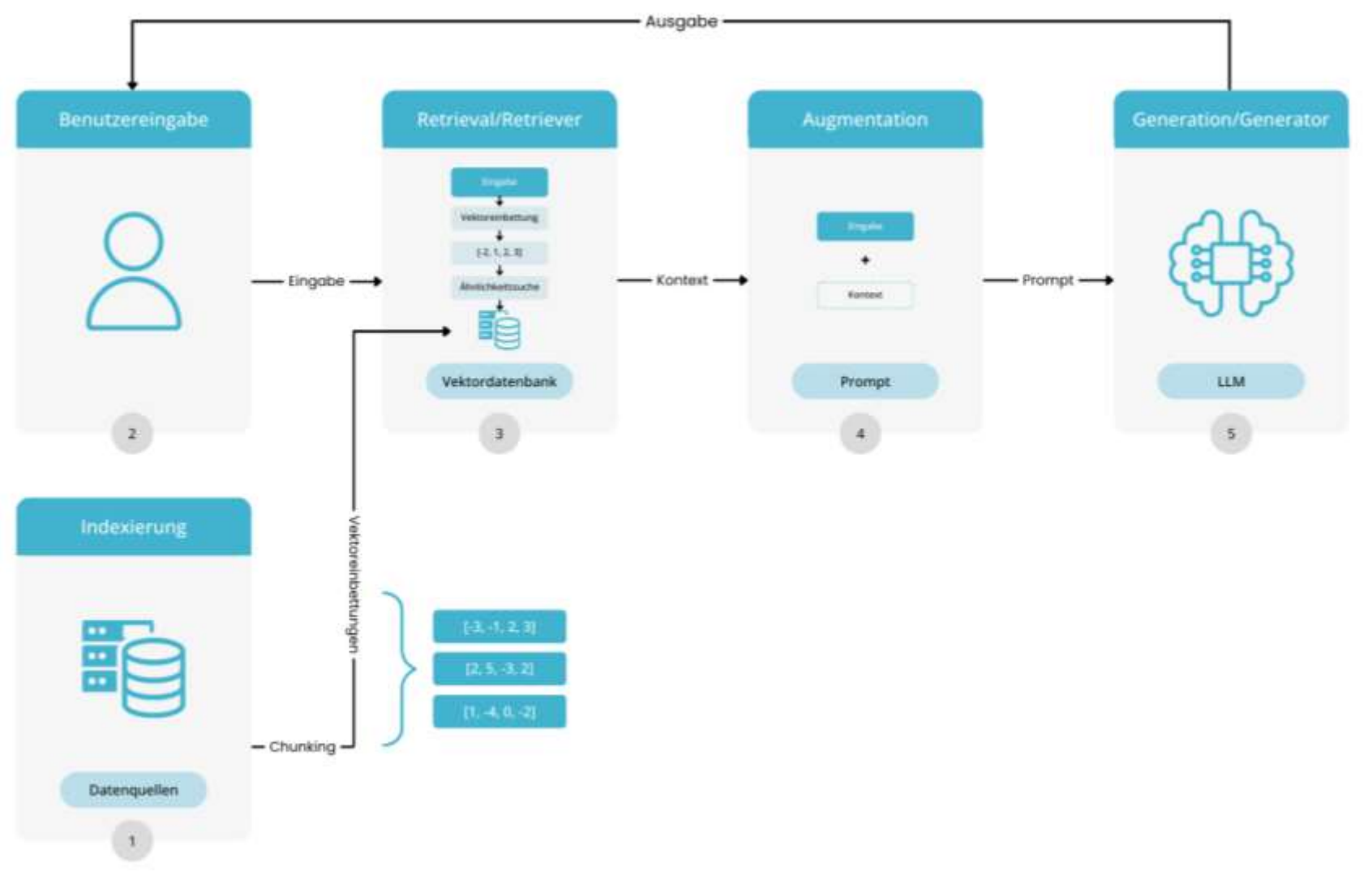
Retrieval Augmented Generation (RAG)

"RAG is like pointing a large language model in the right direction and giving it more specified guidance.“

Gabriel Skelton, Head of Artificial Intelligence Solutions & LinkedIn Top AI Voice

- Extends LLM approaches to specific domains and knowledge base
- “Chat with your data”
- Up-to-date information
- More control
- Reduces hallucination
- Combines deep learning with natural language processing

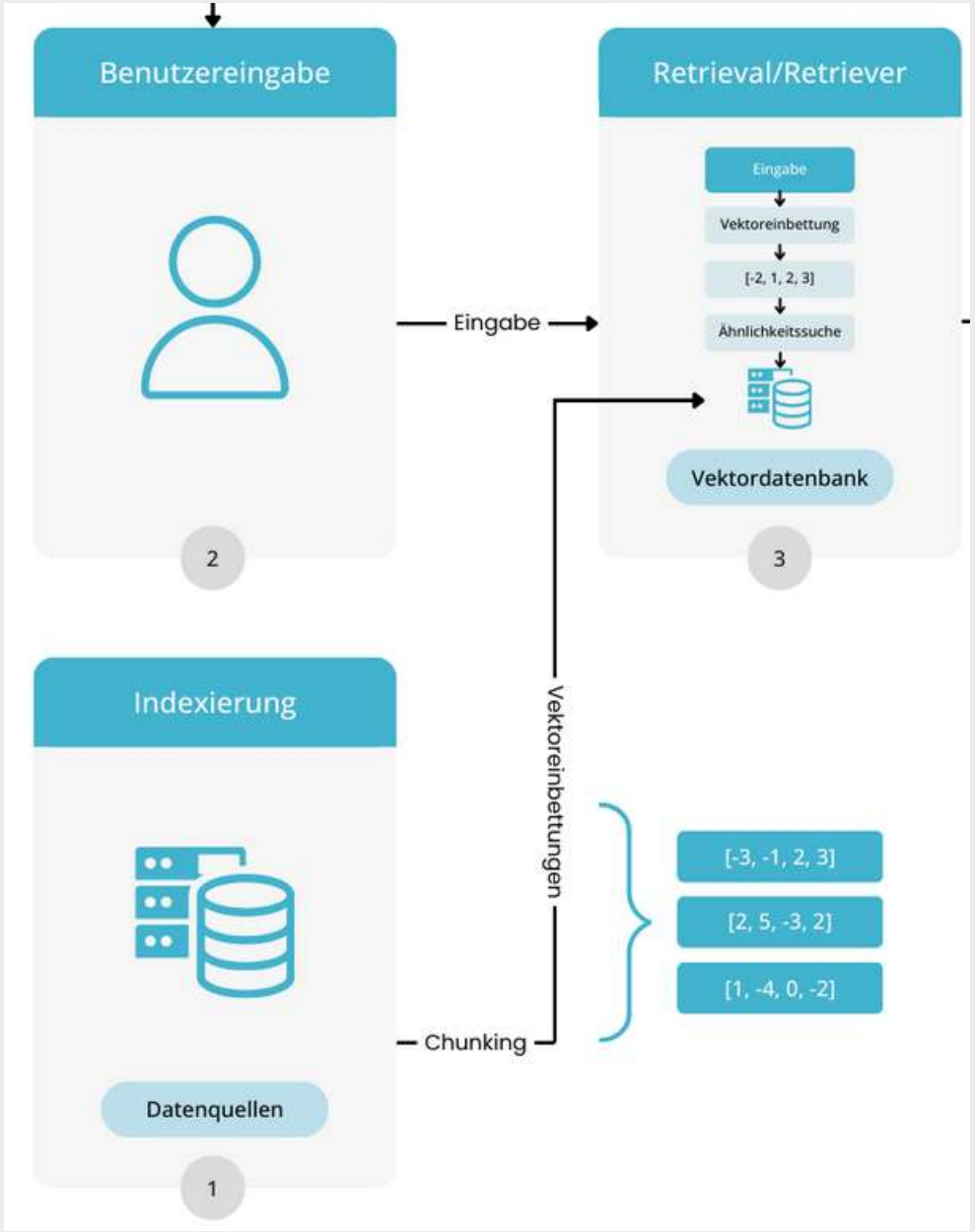
Retrieval-Augmented Generation



Indexing

1. Split into chunks
2. Convert chunks into numerical representation
3. Save vector embeddings in vector database

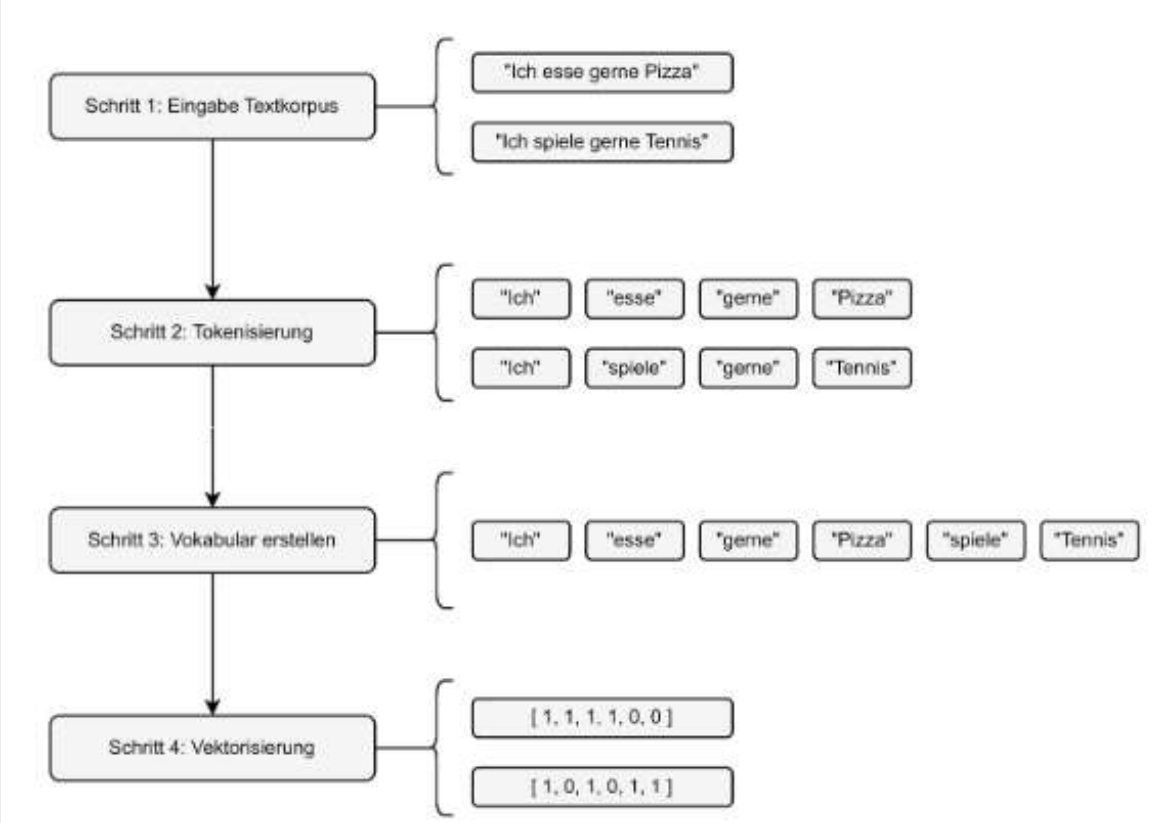
Embedding: Numerical representations of concepts converted to number sequences, high-dimensional



Embeddings: Bag of Words

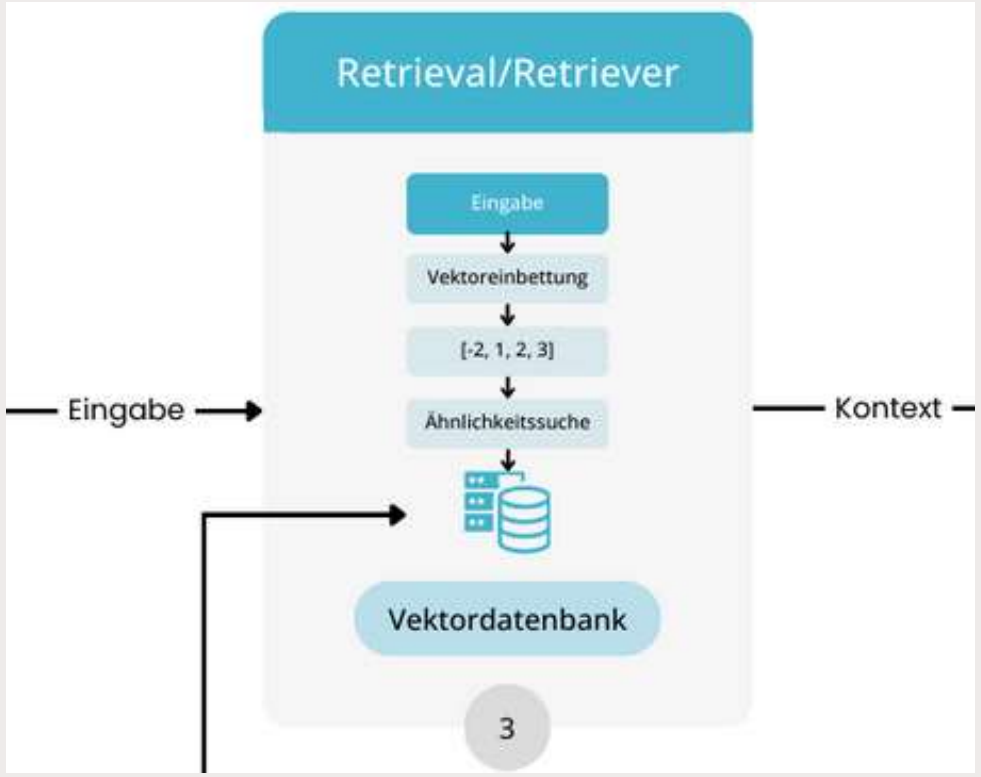
```
import openai
response = openai.Embedding.create(
  input="porcine pals say",
  model="text-embedding-ada-002"
)

print(response)
{
  "data": [
    {
      "embedding": [-0.0108, -0.0107, 0.0323,...,-0.0114],
      "index": 0,
      "object": "embedding"
    }
  ],
  "model": "text-embedding-ada-002"
}
```



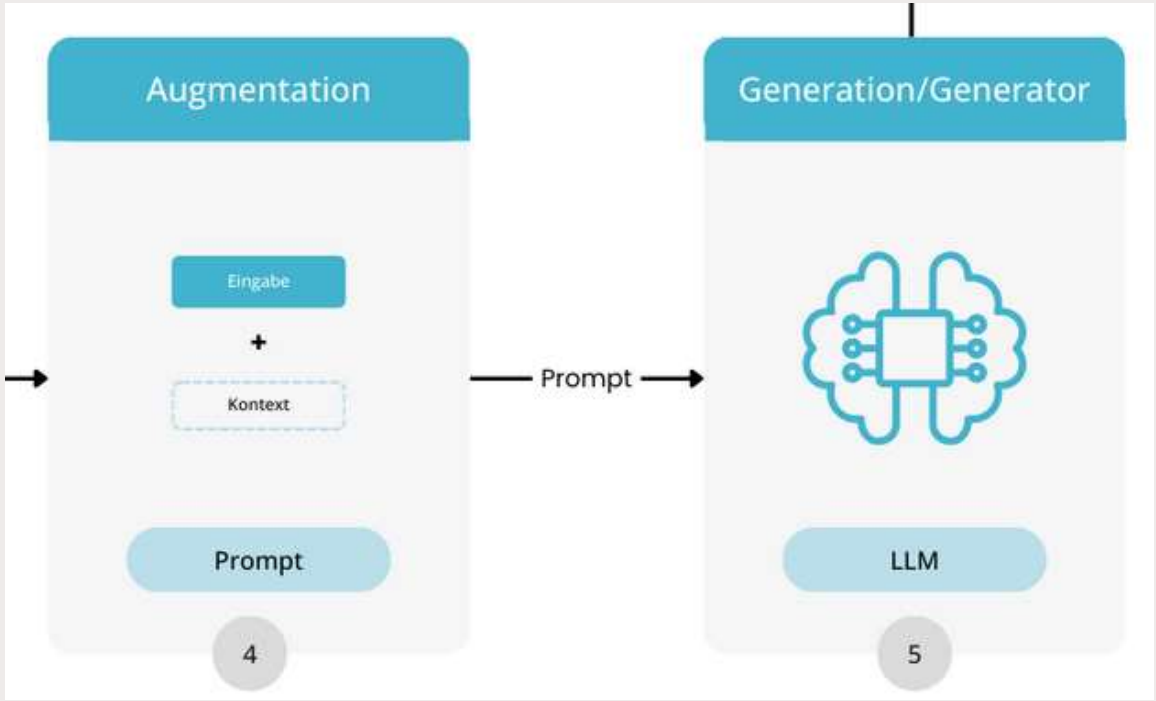
Retrieval

1. Input → Embedding
2. Optional pre-retrieval processing, e. g. enrich with metadata
3. Vector store retrieval: Find k nearest neighbors
4. Optional post-retrieval processing, e. g. summarization of chunks
5. Pass on as context



Augmentation & Generation

1. Generated context + User input + internal prompt
2. Passed on to Generator (LLM / SLM)
3. Language Model generates final answer



Evaluation

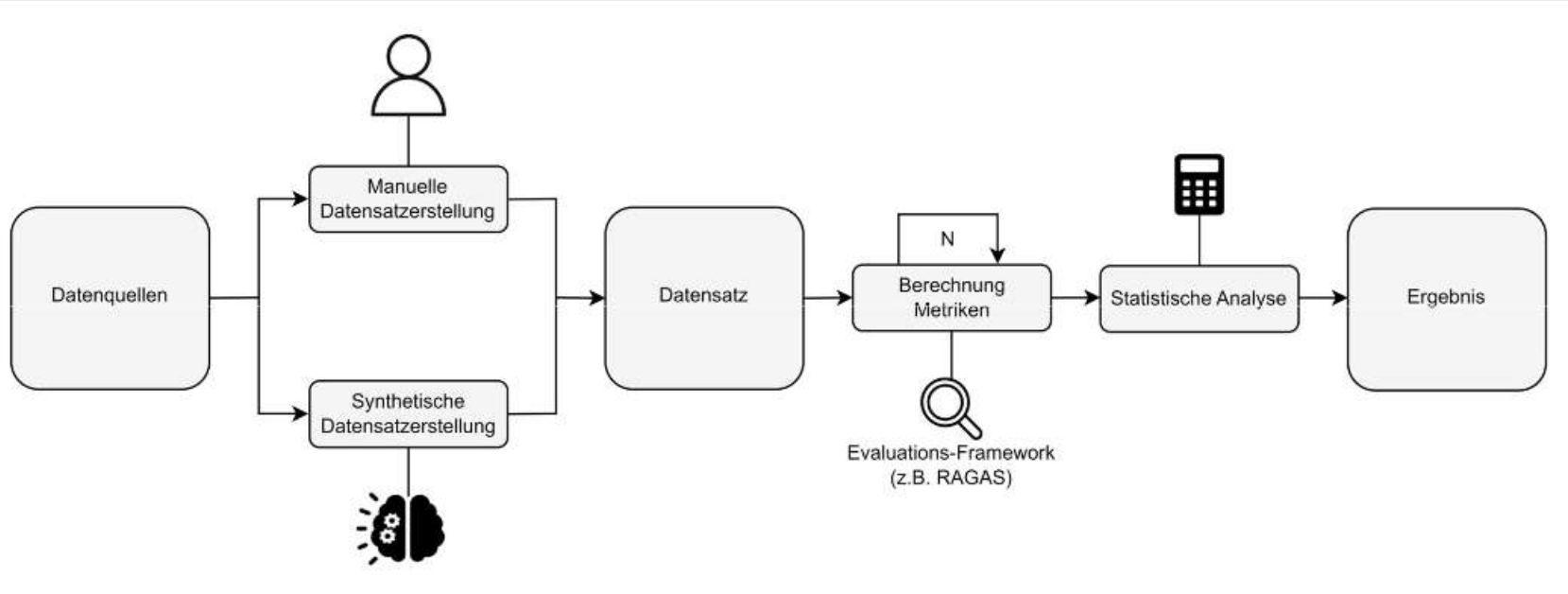
Context Precision

Context Recall

Faithfulness

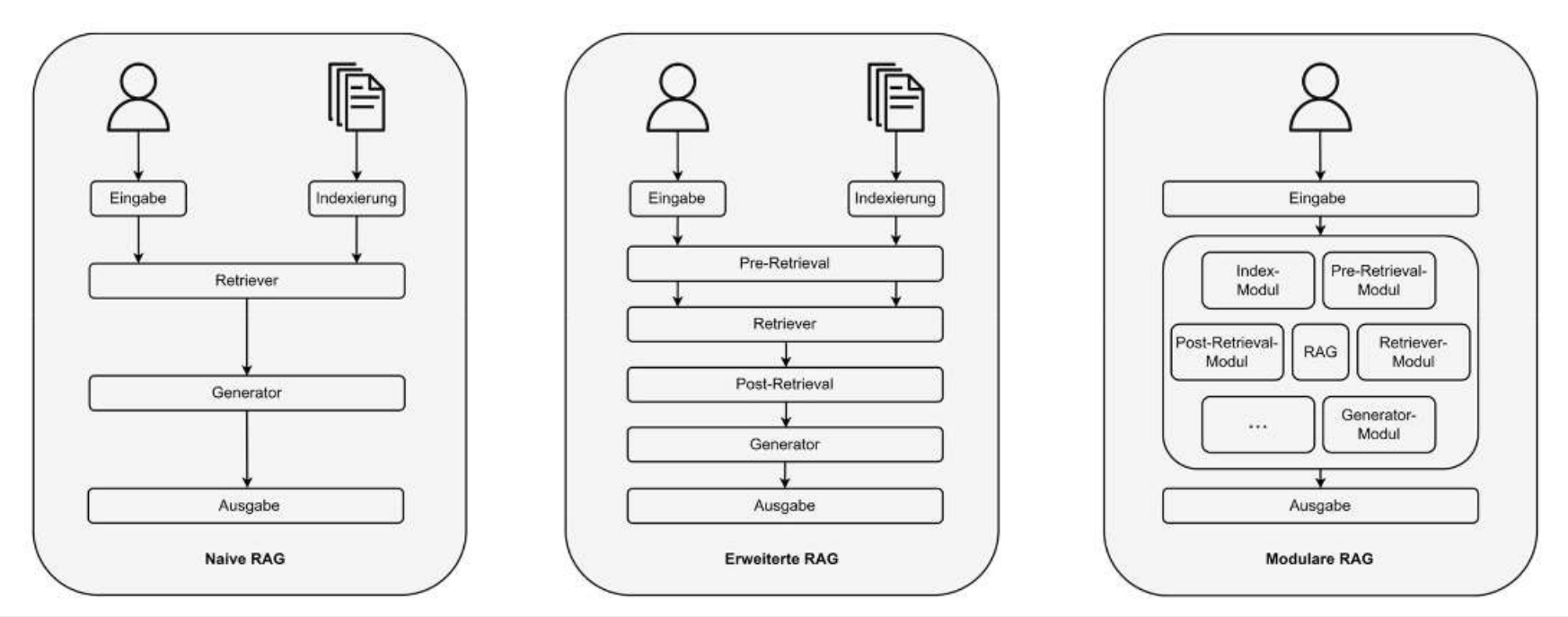
Answer Relevancy

Ragas Score



https://www.ragas.io/

RAG Variations



Retrieval-Augmented Generation

Up-to-date information:

New documents can easily be added to vector database. More relevant and useful responses.

Increased transparency:

Answers are linked to the data source.

Reduced hallucinations:

Reduces hallucinations. Give standardized answer if knowledge is missing.

Reduced costs:

No repeated training needed for LLM.

Code Example

```
# Define loader and get content from document
loader = TextLoader("fc_augsburg.txt")
content = loader.load()

# Define text splitter and split content into chunks
text_splitter = CharacterTextSplitter(separator="\n\n",
   chunk_size=500, chunk_overlap=100)

chunks = text_splitter.split_documents(content)

# Create vector store using chunks and embedding model
vector_store = Chroma.from_documents(
   documents=chunks,
   embedding=FastEmbedEmbeddings(model_name="BAAI/bge-
small-en-v1.5"))

# Define LLM and prompt
llm = ChatOpenAI(model="gpt-3.5-turbo-0125")

prompt = PromptTemplate.from_template(
   """
   Answer the question based only on the following context.
   If you don't know the answer, just say that you don't know.
   Question: {question}
   Context: {context}
   Answer:
   """
)

# Define vector store as the retriever
retriever = vector_store.as_retriever()

rag_chain = (
   {"context": retriever, "question": RunnablePassthrough()}
   | prompt
   | llm
   | StrOutputParser()
)

rag_chain.invoke("Wie viele Mitglieder hat der FC Augsburg?")
# Answer: Der FC Augsburg hat 24.435 Mitglieder
```

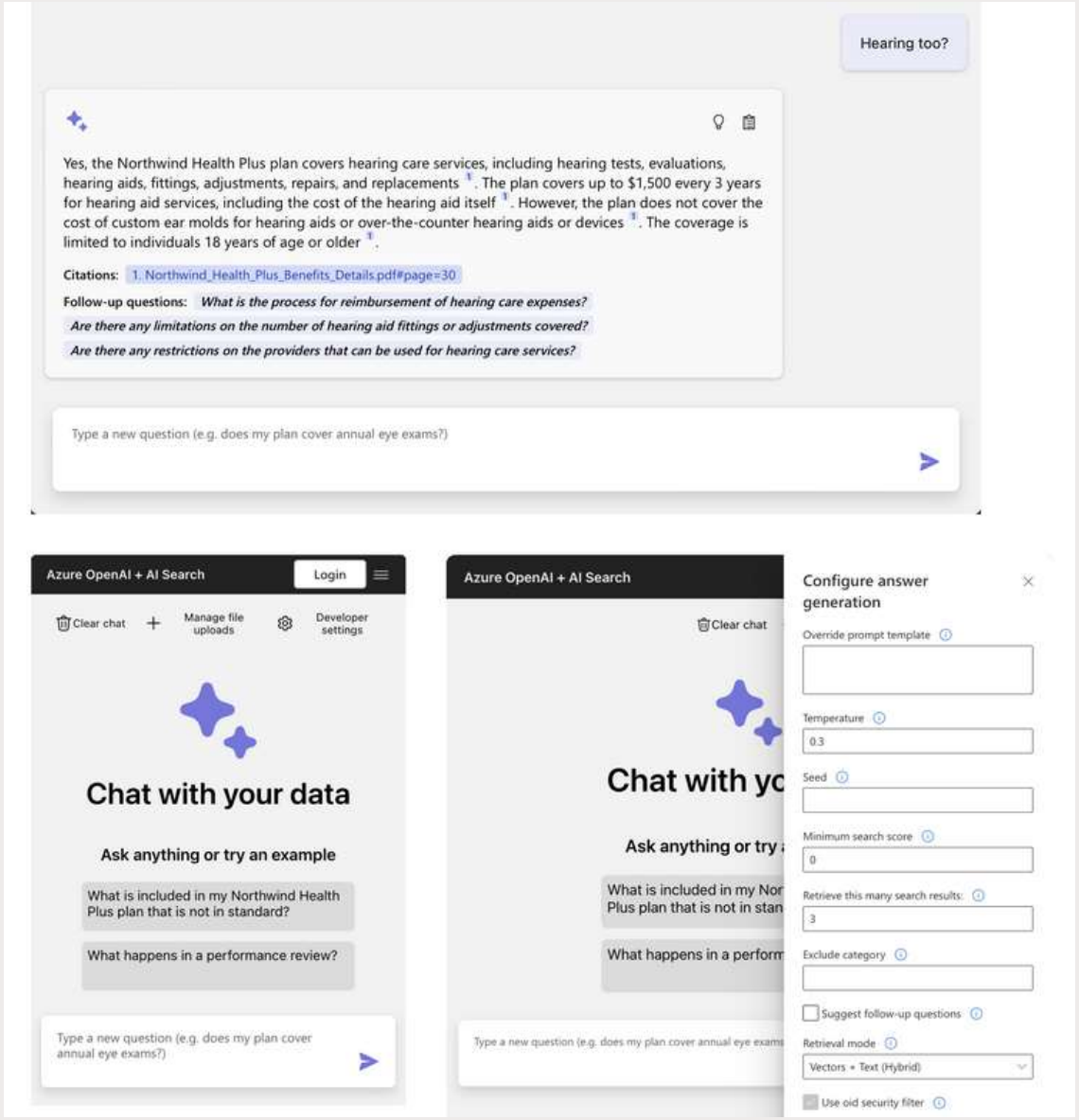
Real-world Applications of RAG Models

1. Advanced Question-Answering Systems
2. Automated Content Creation
3. Summarization of Documents, Meetings, and Articles
4. Conversational Agents and Chatbots, e.g. Customer Service
5. Information Retrieval
6. Educational Tools and Resources
7. Legal Research and Analysis
8. Academic Literature Analysis
9. Content Recommendation Systems
10. Education and E-Learning
11. Enterprise Knowledge Management

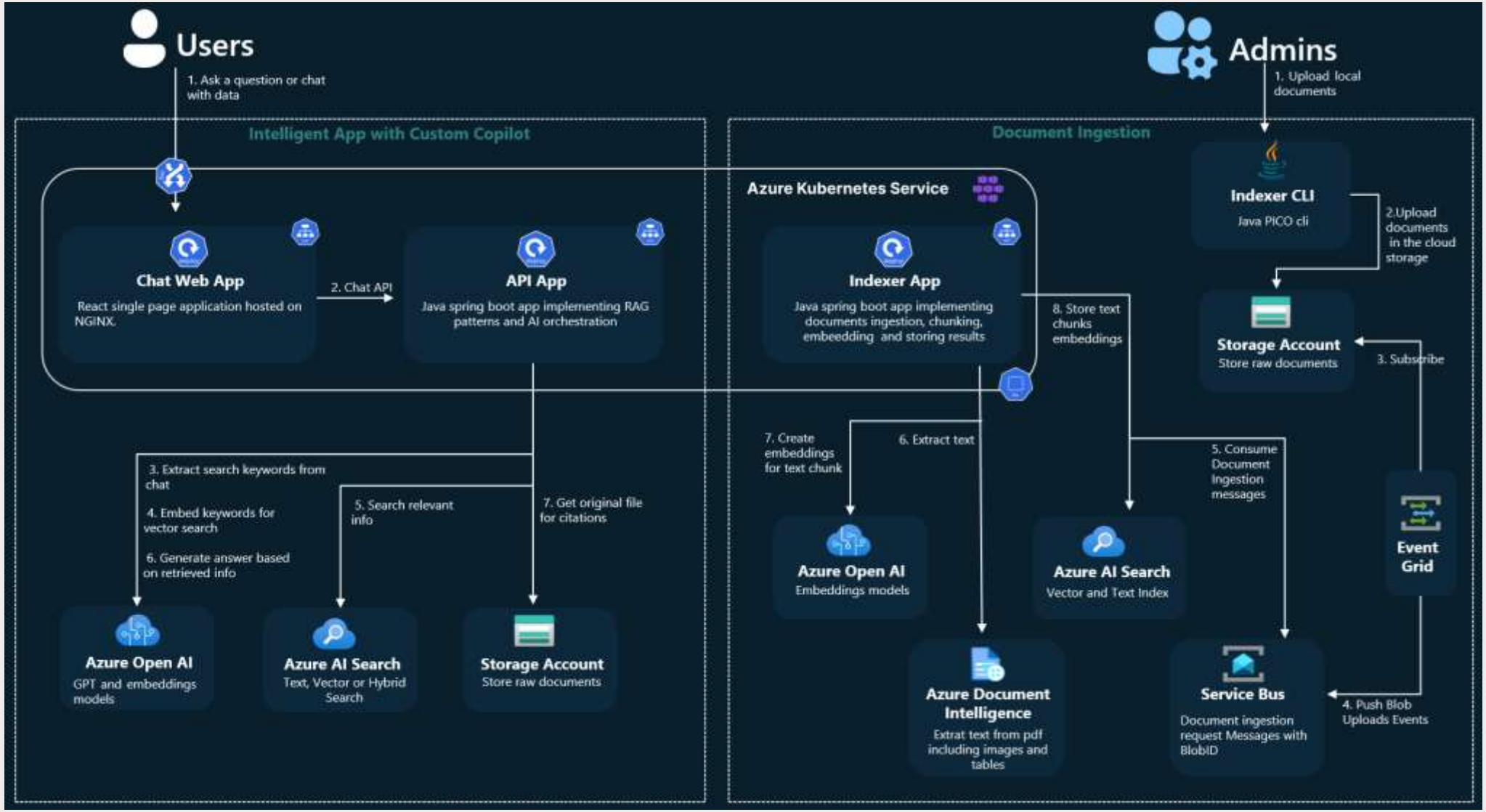
RAG Starter Application

https://github.com/Azure-Samples/azure-search-openai-demo

- Built on Azure resources
- Available in Python, C# and Java
- App Service or Microservice event-driven (Service Bus, Event Grid)
- What's missing:
  - Monitoring
  - Evaluation https://github.com/Azure-Samples/ai-rag-chat-evaluator
  - Admin interface



RAG Starter Application: Java Deployment



CONTACT ME

Dr. Matthias Sommer

AraCom IT Services GmbH
Daimlerstraße 13
86368 Gersthofen

matthias.sommer@aracom.de

Talk to me if you are looking for an industry partner for your student thesis.

Connect on LinkedIn

Open discussion

Have you used LLMs or RAGs?

How is your experience?

Are you still excited?

References

1.Master thesis, Development of a framework for the integration of large language models, domain data and business logic, Andreas Schmid, 2024

2.What is RAG and how will it impact your adoption of AI?

3.Zhao P., Zhang H., Yu Q., Wang Z., Fu F., Yang L., Zhang W., Cui B., (2024). Retrieval-Augmented Generation for AI-Generated Content: A Survey

4.Top 10 Generative AI Tools You Should Know for 2024

5.Amaratunga T., (2023). Understanding Large Language Models

6.Large language models: The basics and their applications

7.Was ist Retrieval Augmented Generation?

8.Retrieval Augmented Generation (RAG) in Azure KI Search

9.Harnessing the power of Large Language Models: A comparative overview of LangChain, Semantic Kernel, AutoGen and more

10.Introducing Code Llama, a state–of–the-art large language model for coding

11.Phi-2: The surprising power of small language models