Generative AI in Cybersecurity

Module 2B: Retrieval Augmented Generation

Henning Thomsen

hth@ucn.dk

5. May 2025

Pba IT-security @ UCN

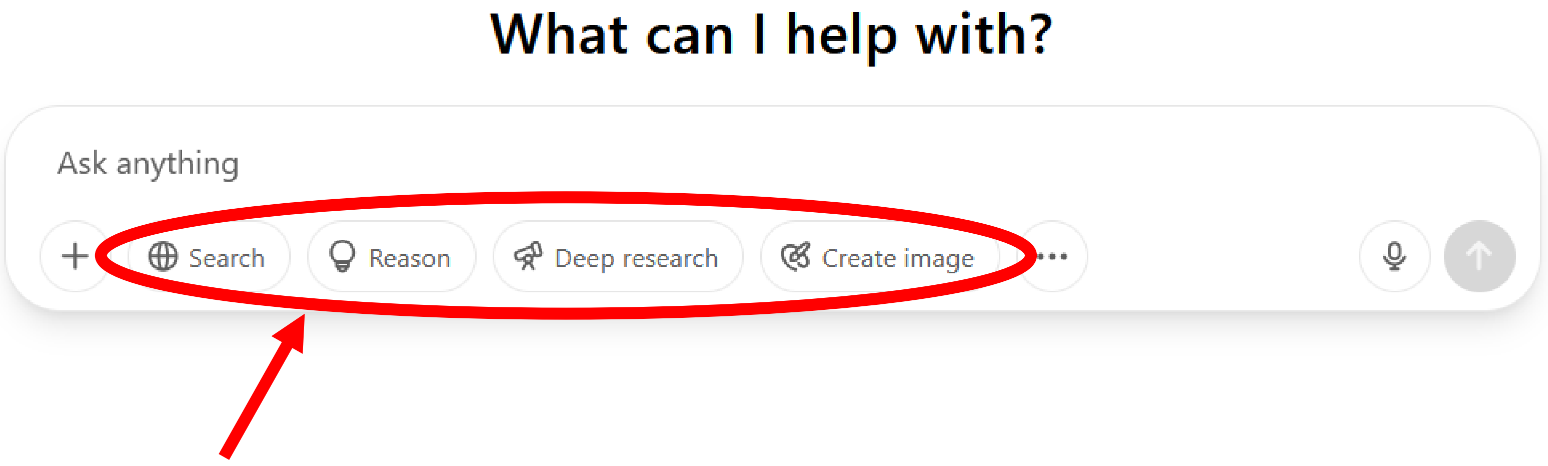
Agenda

- Retrieval Augmented Generation (RAG)
  - Document loading
  - Chunking
  - Vector stores
  - Document querying

Retrieval Augmented Generation

Augment LLM with (relevant) results from a database

ChatGPT

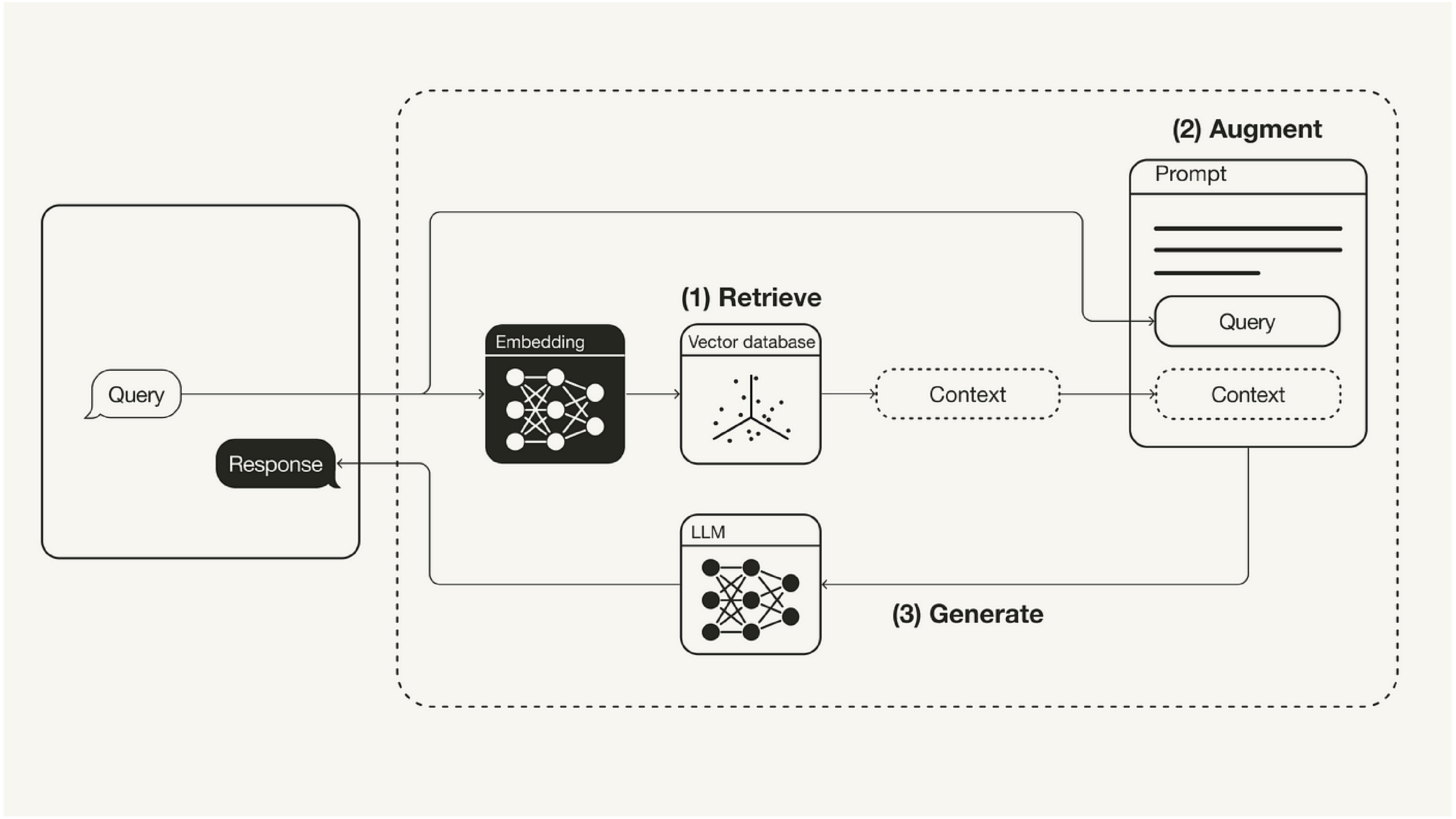


What can I help with?

Ask anything

Extra **tools** added

What is RAG?


(2) Augment
Prompt
Query
Context
(1) Retrieve
Embedding
Vector database
Context
Query
Response
LLM
(3) Generate

Recall spam example

```
prompt_template = """
Classify the given email message as either spam or legitimate.

Examples are given below:

Message: "Hi Alex, just confirming our meeting tomorrow at 10 AM—let me
know if anything changes."
Classification: Legitimate

Message: "Your account has been compromised—click here immediately to
verify your identity and avoid suspension!"
Classification: Spam

Message: {message}
Classification:
"""
```

Limitations

- Context length related to token cost
- Limited context length (is it sufficient)?
- **Size of context vs. maximum context window**

What is RAG?

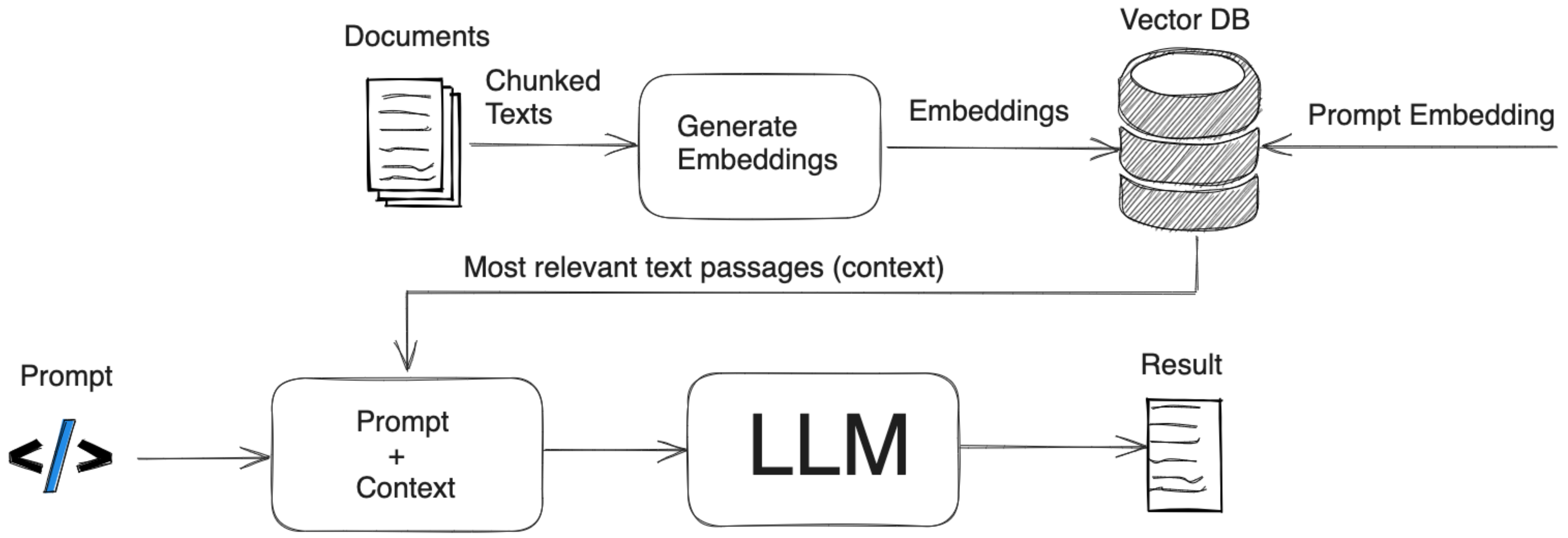


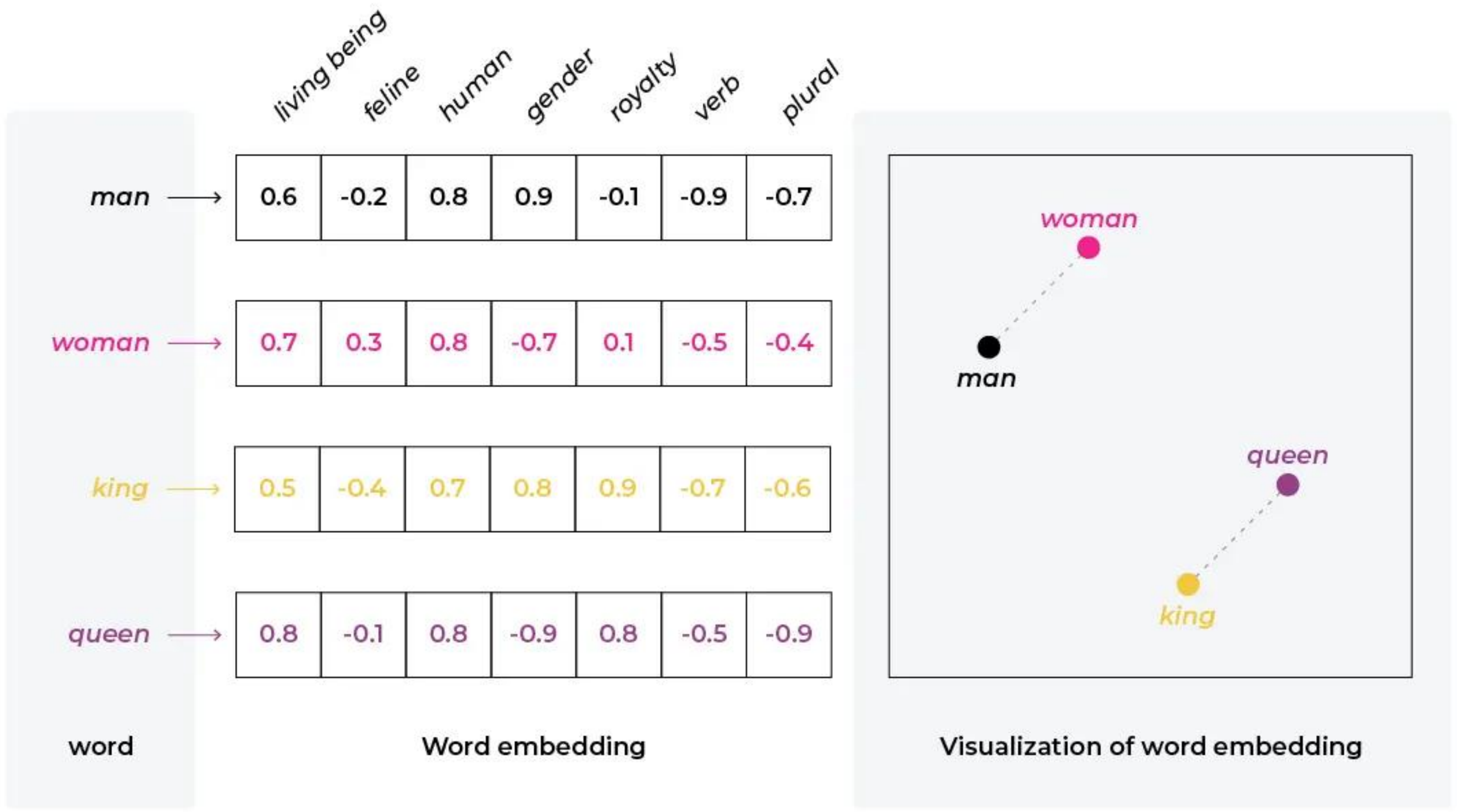
Figure from https://safjan.com/understanding-retrieval-augmented-generation-rag-empowering-llms/

Vector embeddings

- LLMs only accept floating point values as input
  - At the most basic level

- Want to convert our context (data) to these vectors
  - Could be: TXT, PDF, DOC, XLS etc.

Vector embeddings



word	living being	feline	human	gender	royalty	verb	plural
man	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman	0.7	0.3	0.8	-0.7	0.1	-0.5	-0.4
king	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

Figure from https://arize.com/blog-course/embeddings-meaning-examples-and-how-to-compute/

Indexing a vector database

- Uses an **embedding** to map documents to vectors
    - OpenAIEmbeddings
    - OllamaEmbeddings

- Need to choose among vector stores
    - Chroma
    - FAISS
    - Qdrant
    - …

Loading documents

```python
from langchain_community.document_loaders import PyPDFLoader

file_path = (
    "A-Survey-of-Large-Language-Models.pdf"
)
loader = PyPDFLoader(file_path)
pages = loader.load()
```

Splitting documents

```
from langchain_text_splitters import RecursiveCharacterTextSplitter

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=200, add_start_index=True
)
all_splits = text_splitter.split_documents(pages)
```

Demo: https://chunkviz.up.railway.app/

Vector DB ingestion

```
from langchain_openai import OpenAIEmbeddings

from langchain_community.vectorstores import Chroma

vectorstore = Chroma.from_documents(documents=all_splits,
embedding=OpenAIEmbeddings())
```

Querying documents from DB

```
retriever = vectorstore.as_retriever(search_type="similarity",
search_kwargs={"k": 6})

retrieved_docs = retriever.invoke("What are neural language models?")
```

A simple RAG application using LCEL

```
vectorstore = Chroma.from_documents(documents=all_splits,
embedding=OpenAIEmbeddings())
retriever = vectorstore.as_retriever()

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

result = chain.invoke("What are neural language models?")
```