IESTI05 – Edge AI

Machine Learning System Engineering

20. SLMs: Optimization Techniques - RAG

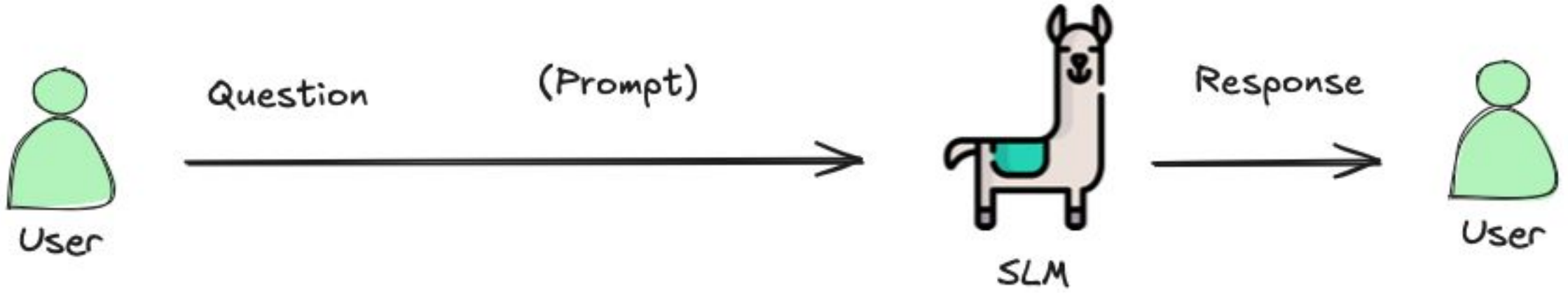
UNIFEI

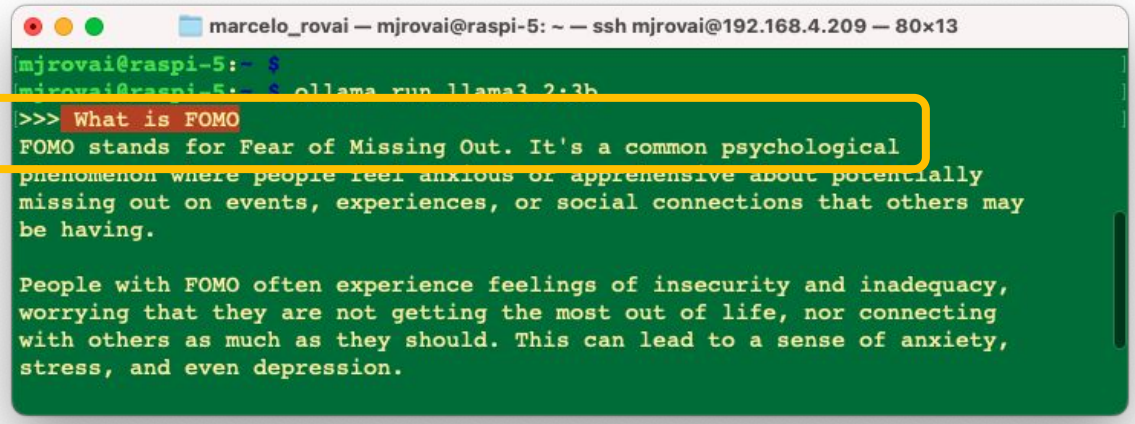
Marcelo Rovai

Professor Honoris Causa

Retrieval-Augmented Generation (RAG)

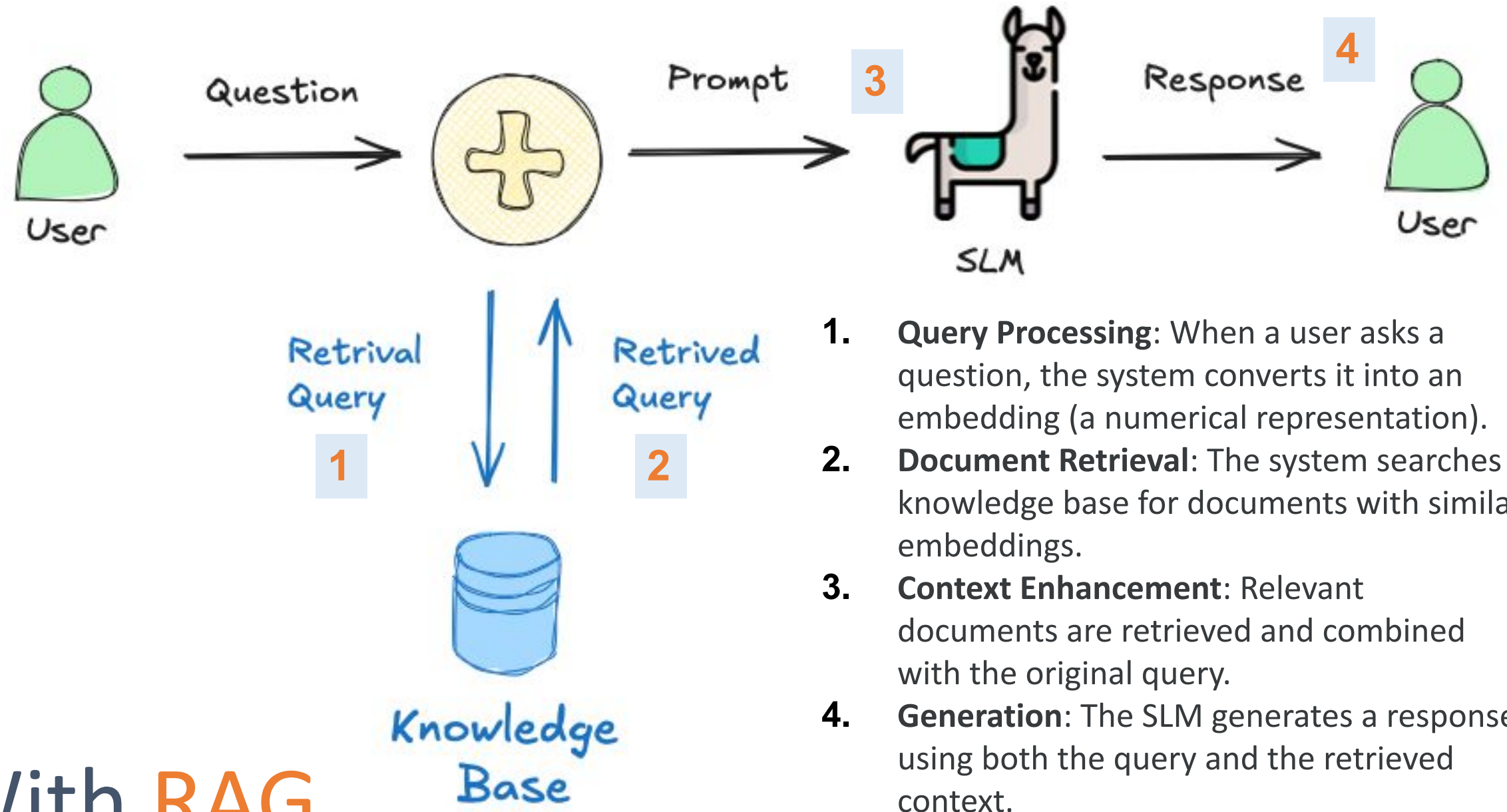
“A method created by the FAIR team at Meta to enhance the accuracy of Large Language Models (LLMs) and reduce false information or “hallucinations.”





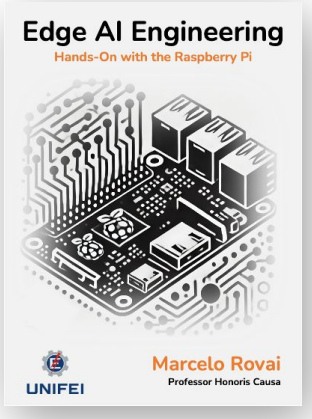
Usual Prompt

With RAG



1. **Query Processing**: When a user asks a question, the system converts it into an embedding (a numerical representation).
2. **Document Retrieval**: The system searches a knowledge base for documents with similar embeddings.
3. **Context Enhancement**: Relevant documents are retrieved and combined with the original query.
4. **Generation**: The SLM generates a response using both the query and the retrieved context.

With RAG



Knowledge Base

```
marcelo_rovai — mjrovai@raspi-5-SD: ~/Documents/OLLAMA/RAG

Your question: What is FOMO?

Generating answer...

Question: What is FOMO?
Retrieving documents...
Retrieved 2 document chunks
Generating answer...
Response latency: 57.84 seconds using model: llama3.2:3b

ANSWER:
==================================================
FOMO stands for Faster Objects, More Objects.
==================================================

Your question:
```

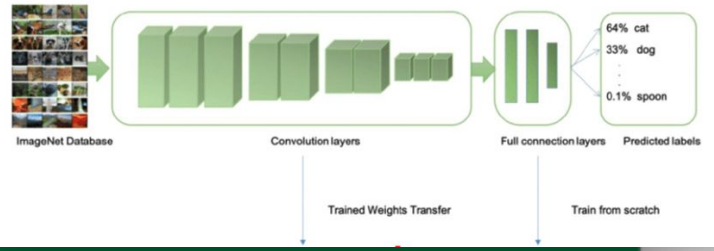
Training a FOMO Model at Edge Impulse Studio

The inference with the SSD MobileNet model worked well, but the latency was significantly high. The inference varied from 0.5 to 1.3 seconds on a Raspi-Zero, which means around or less than 1 FPS (1 frame per second). One alternative to speed up the process is to use FOMO (Faster Objects, More Objects).

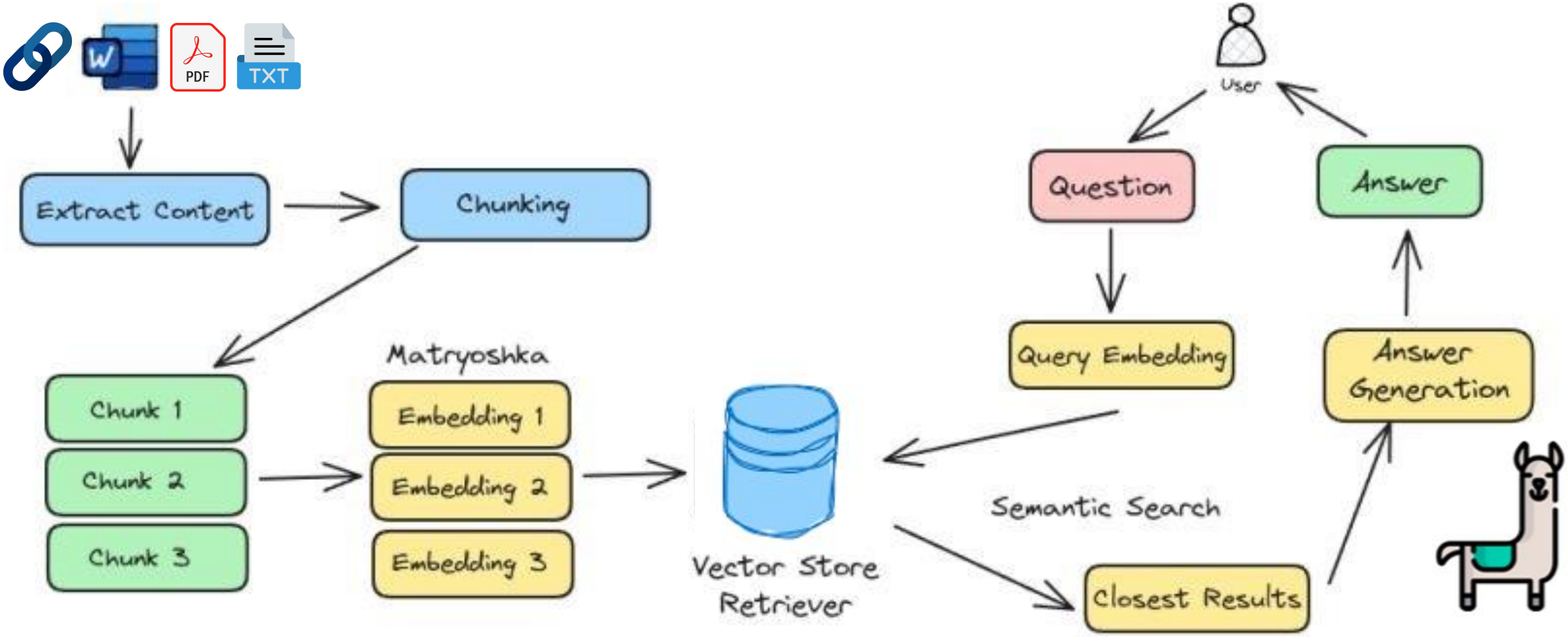
This novel machine learning algorithm lets us count multiple objects and find their location in an image in real-time using up to 30x less processing power and memory than MobileNet SSD or YOLO. The main reason this is possible is that while other models calculate the object's size by drawing a square around it (bounding box), FOMO ignores the size of the image, providing only the information about where the object is located in the image through its centroid coordinates.

How FOMO works?

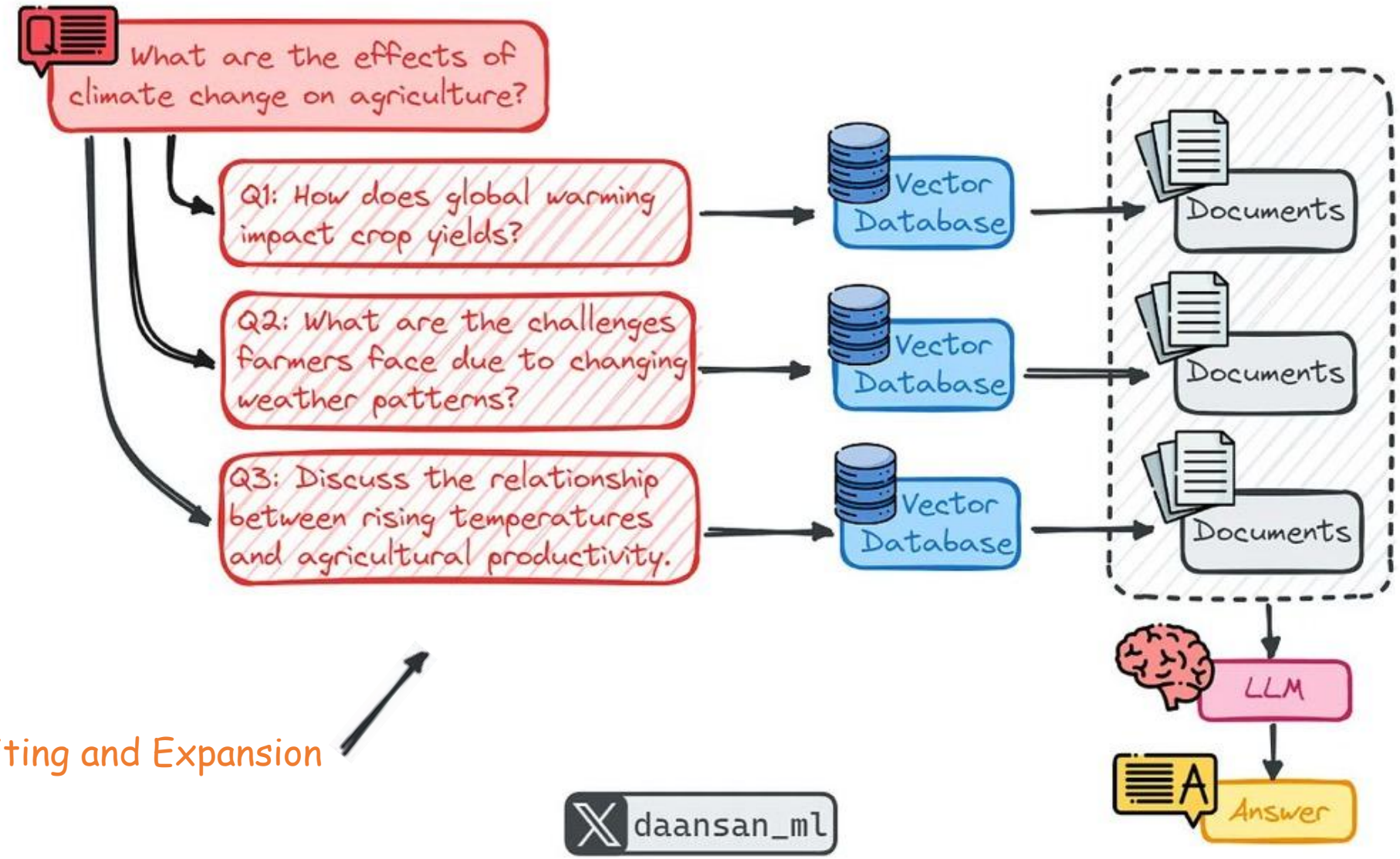
In a typical object detection pipeline, the first stage is extracting features from the input image. **FOMO leverages MobileNetV2 to perform this task**. MobileNetV2 processes the input image to produce a feature map that captures essential characteristics, such as textures, shapes, and object edges, in a computationally efficient way.



RAG: Simple Query (“Naive RAG”)



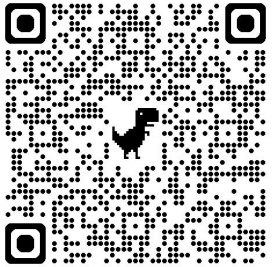
Advanced RAG: Multi Query



Query Rewriting and Expansion

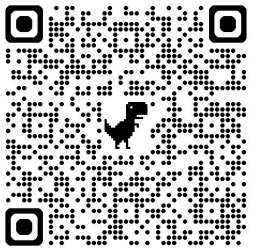
Simple RAG example

40-RAG-simple-bee.ipynb

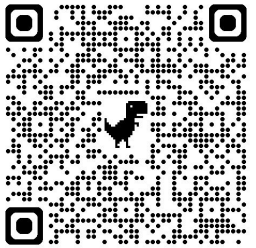


Implementing a Naive RAG System

1. **Creating the Vector Database** (10-Create-Persistent-Vector-Database.py).
   This script builds a knowledge base by:
   - Loading documents from PDFs and URLs
   - Splitting them into manageable chunks
   - Creating embeddings for each chunk
   - Storing these embeddings in a vector database (Chroma)



2. **Querying the Database** (20-Query-the-Persistent-RAG-Database.py).
   This script:
   - Loads the saved vector database
   - Accepts user queries
   - Retrieves relevant documents based on query similarity
   - Combines documents with the query in a prompt
   - Generates a response using the SLM



Book: Advanced EdgeAI: RAG

Naive RAG Setup

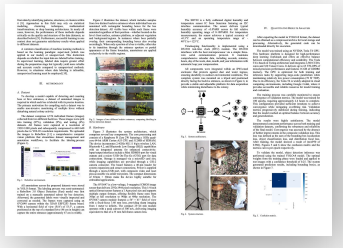
Inside the env:

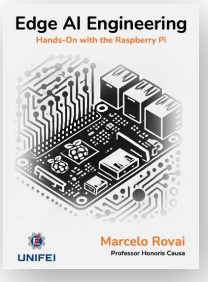
```
source ~/ollama/bin/activate
```

Install:

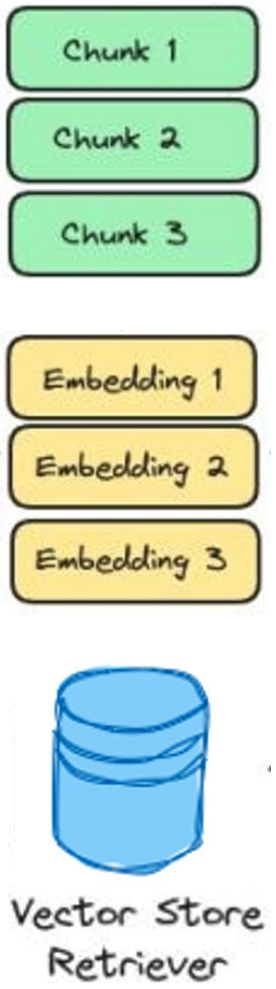
```
- pip install -U 'langchain-chroma'
- pip install -U langchain
- pip install -U langchain-community
- pip install -U langchain-ollama
- pip install -U langchain-text-splitter
- pip install -U langchain-community pypdf
- pip install tiktoken
- pip install -U langsmith
```

Create Knowledge Base





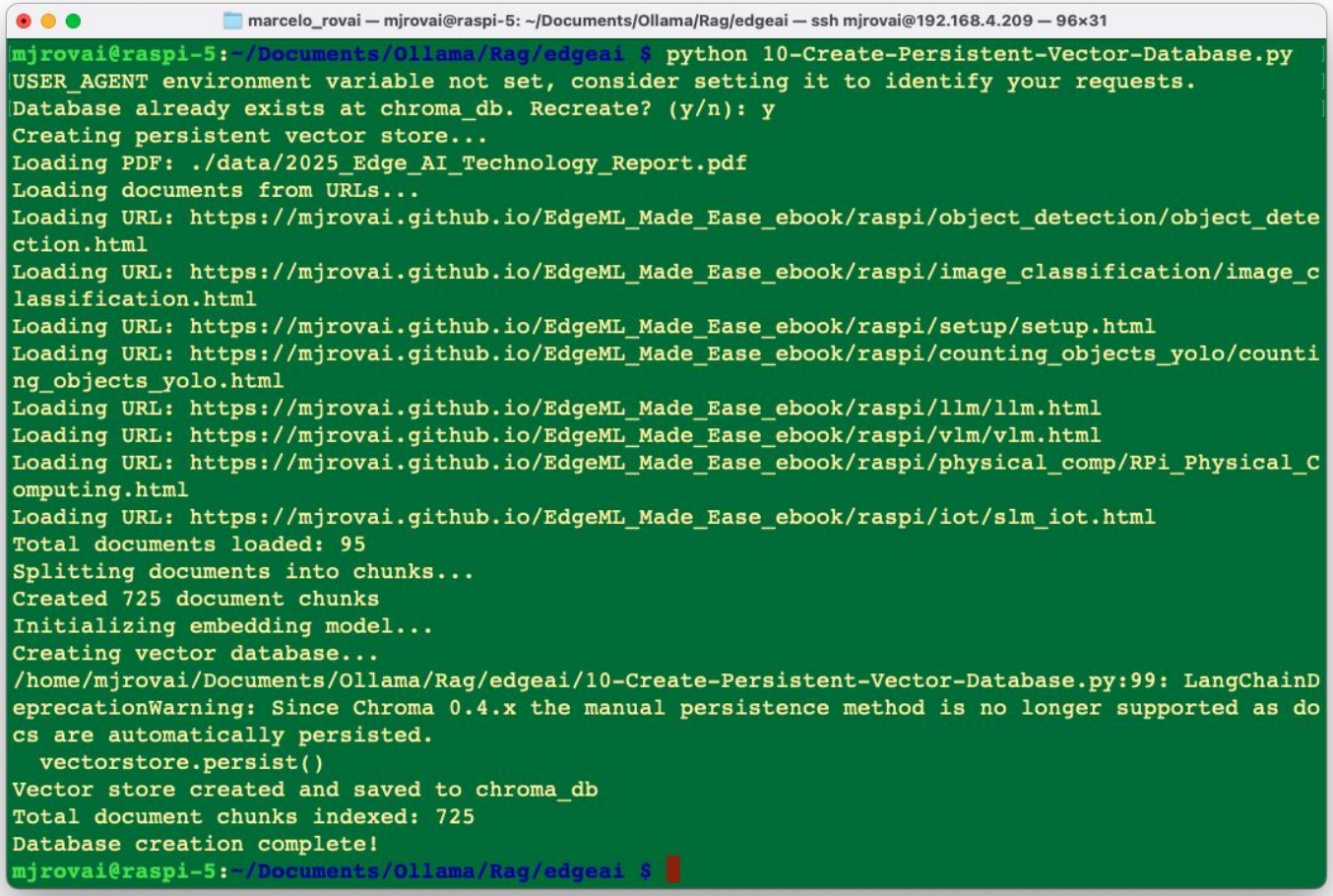

```python
import warnings
import os
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import WebBaseLoader
from langchain_community.document_loaders import PyPDFLoader
from langchain_ollama import OllamaEmbeddings
from langchain_community.vectorstores import Chroma

# Suppress LangSmith warnings
warnings.filterwarnings("ignore",
                        message="API key must be provided when using hosted LangSmith API",
                        category=UserWarning)


# Vector Database

# Define persistent directory for Chroma
PERSIST_DIRECTORY = "chroma_db"

# PDF documents to include
pdf_paths = ["./Use_Edge_ML_Non-Invasive_Beehive_Monitoring_System.pdf"]


# Define URLs for document sources
urls = [
    "https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/setup/setup.html",
    "https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/object_detection/object_detection_fundamentals.html",
    "https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/object_detection/cv_yolo.html",
    "https://mjrovai.github.io/EdgeML_Made_Ease_ebook/raspi/llm/llm.html",
]
```

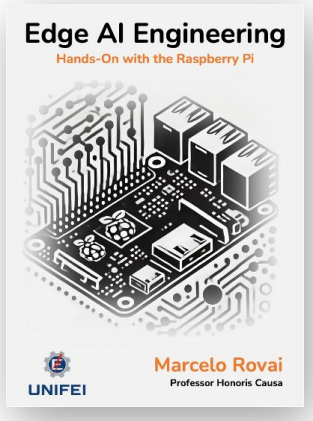



```python
# Split documents
print("Splitting documents into chunks...")
text_splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
    chunk_size=300, chunk_overlap=30
)
doc_splits = text_splitter.split_documents(docs_list)
print(f"Created {len(doc_splits)} document chunks")

# Create embedding function
print("Initializing embedding model...")
embedding_function = OllamaEmbeddings(model="nomic-embed-text")

# Create and persist vectorstore to disk
print("Creating vector database...")
vectorstore = Chroma.from_documents(
    documents=doc_splits,
    collection_name="rag-edgeai-eng-chroma",
    embedding=embedding_function,
    persist_directory=PERSIST_DIRECTORY
)
```

Create Knowledge Base

```
python 10-Create-Persistent-Vector-Database.py
```

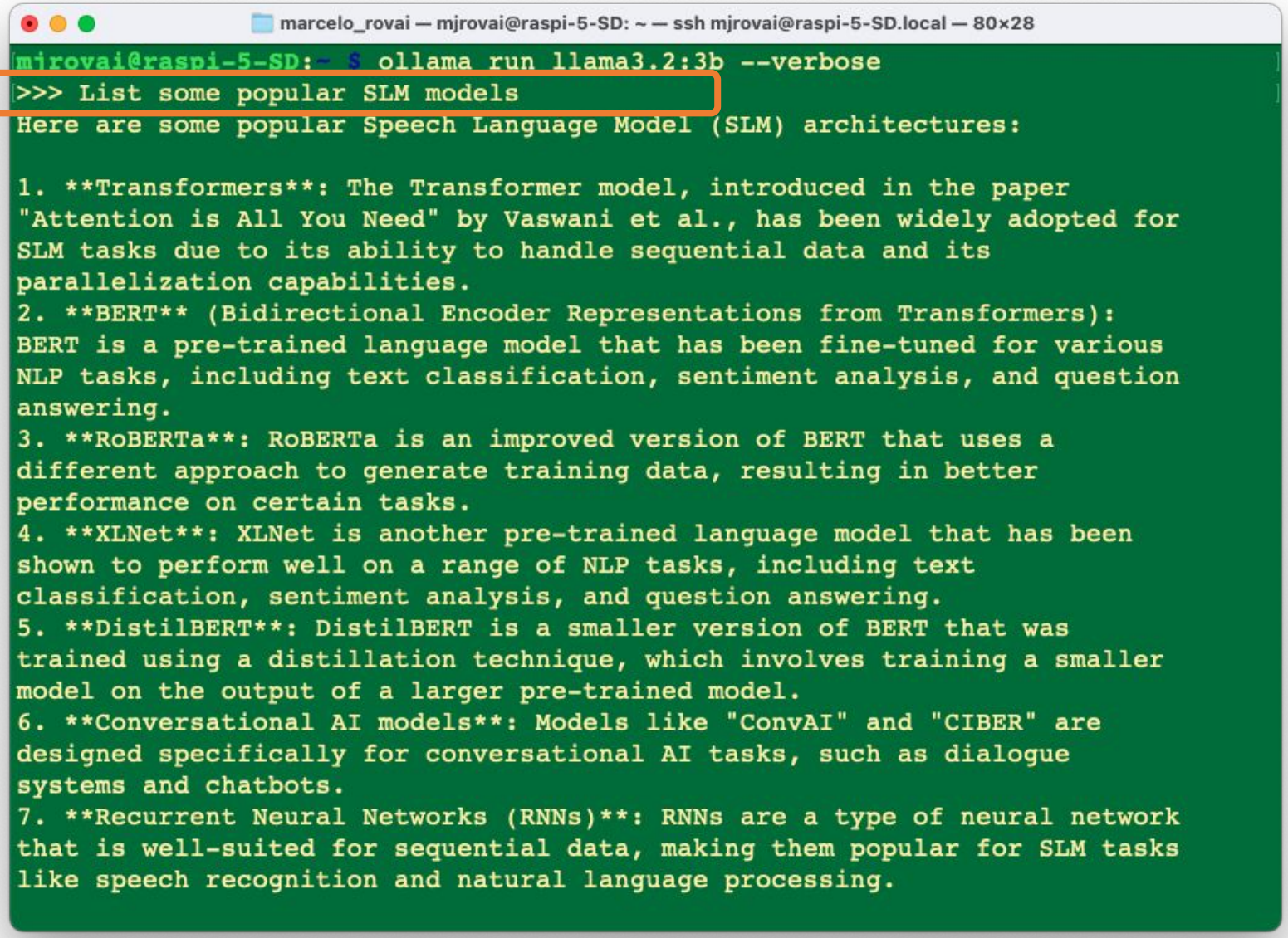


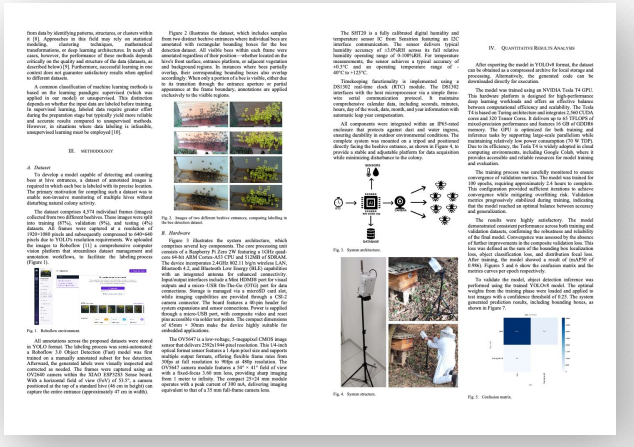


Knowledge Base

```
marcelo_rovai — mjrovai@raspi-5-SD: ~/Documents/OLLAMA/RAG-Naive — ssh mjrovai@raspi-5-SD.local — 80×30
mjrovai@raspi-5-SD:~ $ source ~/ollama/bin/activate
(ollama) mjrovai@raspi-5-SD:~ $ cd Documents/
(ollama) mjrovai@raspi-5-SD:~/Documents $ cd OLLAMA/
(ollama) mjrovai@raspi-5-SD:~/Documents/OLLAMA $ cd RAG-Naive/
(ollama) mjrovai@raspi-5-SD:~/Documents/OLLAMA/RAG-Naive $ python 20-Query-the-Persistent-RAG-Database.py
Loading existing vector store...
==== RAG Query System ====
Type your questions and press Enter. Type 'quit' to exit.

Your question: List some popular SLM models

Generating answer...

Question: List some popular SLM models
Retrieving documents...
Retrieved 4 document chunks
Generating answer...
Response latency: 152.53 seconds using model: llama3.2:3b

ANSWER:
==================================================
Some popular Small Language Models (SLMs) include Meta Llama, Microsoft PHI, Google Gemma, Ollama, MoonDream, Lava-Phi-3, and Gemma3. These models are compact versions of larger language models that excel in specific tasks and domains, making them ideal for targeted applications on edge devices. They offer a range of benefits, including reduced parameter count, lower memory footprint, faster inference time, energy efficiency, and offline functionality.
==================================================
```

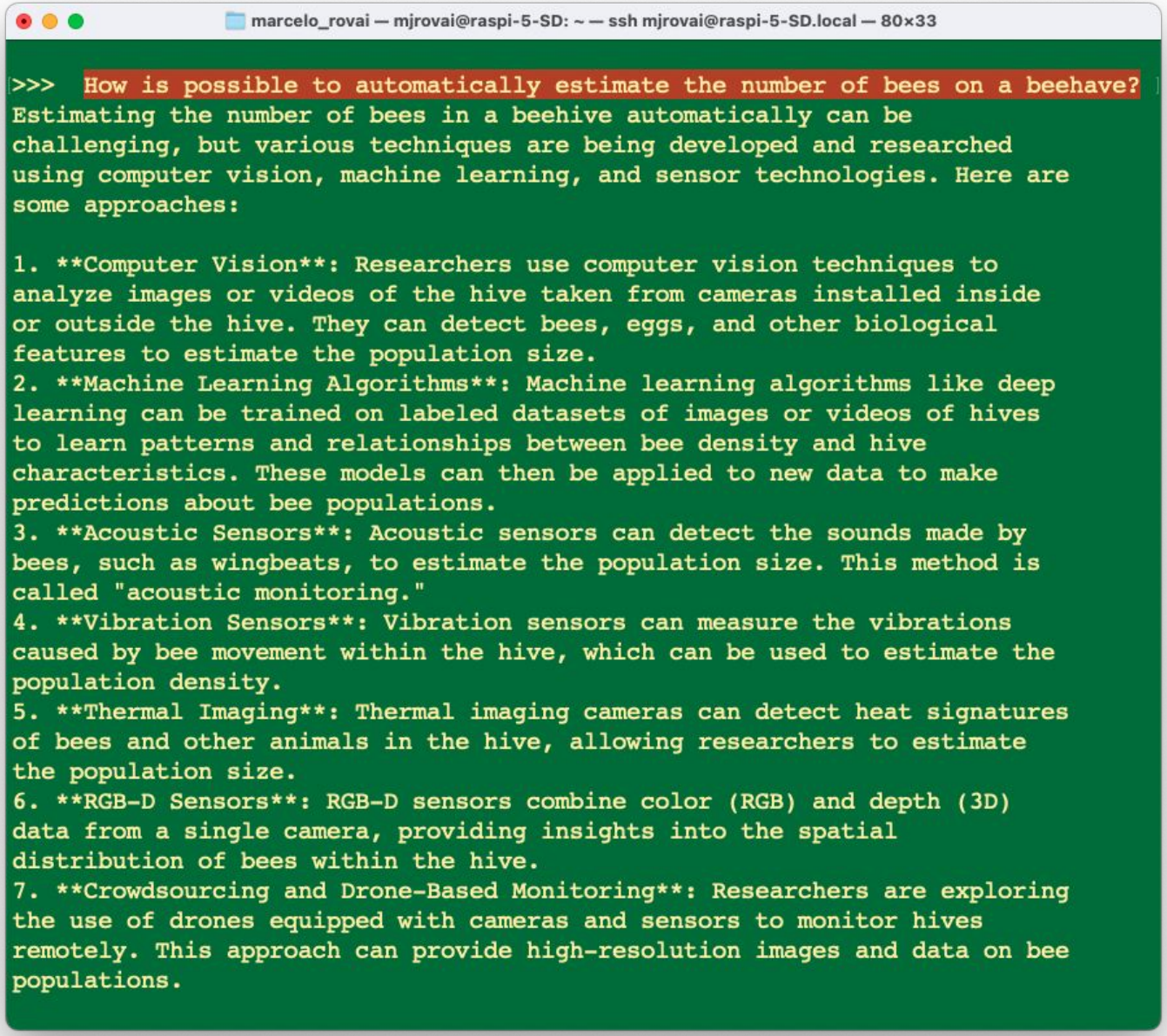
No RAG


marcelo_rovai — mjrovai@raspi-5-SD: ~ — ssh mjrovai@raspi-5-SD.local — 80×28
mjrovai@raspi-5-SD:~ $ ollama run llama3.2:3b --verbose
>>> List some popular SLM models
Here are some popular Speech Language Model (SLM) architectures:
1. **Transformers**: The Transformer model, introduced in the paper "Attention is All You Need" by Vaswani et al., has been widely adopted for SLM tasks due to its ability to handle sequential data and its parallelization capabilities.
2. **BERT** (Bidirectional Encoder Representations from Transformers): BERT is a pre-trained language model that has been fine-tuned for various NLP tasks, including text classification, sentiment analysis, and question answering.
3. **RoBERTa**: RoBERTa is an improved version of BERT that uses a different approach to generate training data, resulting in better performance on certain tasks.
4. **XLNet**: XLNet is another pre-trained language model that has been shown to perform well on a range of NLP tasks, including text classification, sentiment analysis, and question answering.
5. **DistilBERT**: DistilBERT is a smaller version of BERT that was trained using a distillation technique, which involves training a smaller model on the output of a larger pre-trained model.
6. **Conversational AI models**: Models like "ConvAI" and "CIBER" are designed specifically for conversational AI tasks, such as dialogue systems and chatbots.
7. **Recurrent Neural Networks (RNNs)**: RNNs are a type of neural network that is well-suited for sequential data, making them popular for SLM tasks like speech recognition and natural language processing.




```
marcelo_rovai — mjrovai@raspi-5-SD: ~/Documents/OLLAMA/RAG-Naive — ssh mjrovai@raspi-5-SD.local — 80...

Your question: How is possible to automatically estimate the number of bees on a
 beehave?

Generating answer...

Question:  How is possible to automatically estimate the number of bees on a bee
have?
Retrieving documents...
Retrieved 4 document chunks
Generating answer...
Response latency: 157.85 seconds using model: llama3.2:3b

ANSWER:
==================================================
The system uses a YOLOv8n object detection model to automatically estimate the n
umber of bees on a beehive. It is deployed on a Raspberry Pi Zero 2W embedded pl
atform, which enables local processing without requiring cloud transmission. The
 system has been shown to have high accuracy and reliability in estimating bee n
umbers, with a mean bias of 3.4 bees between automated and manual counts.
==================================================

Your question:
```

No RAG


marcelo_rovai — mjrovai@raspi-5-SD: ~ — ssh mjrovai@raspi-5-SD.local — 80×33
>>> How is possible to automatically estimate the number of bees on a beehave?
Estimating the number of bees in a beehive automatically can be challenging, but various techniques are being developed and researched using computer vision, machine learning, and sensor technologies. Here are some approaches:
1. **Computer Vision**: Researchers use computer vision techniques to analyze images or videos of the hive taken from cameras installed inside or outside the hive. They can detect bees, eggs, and other biological features to estimate the population size.
2. **Machine Learning Algorithms**: Machine learning algorithms like deep learning can be trained on labeled datasets of images or videos of hives to learn patterns and relationships between bee density and hive characteristics. These models can then be applied to new data to make predictions about bee populations.
3. **Acoustic Sensors**: Acoustic sensors can detect the sounds made by bees, such as wingbeats, to estimate the population size. This method is called "acoustic monitoring."
4. **Vibration Sensors**: Vibration sensors can measure the vibrations caused by bee movement within the hive, which can be used to estimate the population density.
5. **Thermal Imaging**: Thermal imaging cameras can detect heat signatures of bees and other animals in the hive, allowing researchers to estimate the population size.
6. **RGB-D Sensors**: RGB-D sensors combine color (RGB) and depth (3D) data from a single camera, providing insights into the spatial distribution of bees within the hive.
7. **Crowdsourcing and Drone-Based Monitoring**: Researchers are exploring the use of drones equipped with cameras and sensors to monitor hives remotely. This approach can provide high-resolution images and data on bee populations.

Optimized Naïve RAG Query

```
marcelo_rovai — mjrovai@raspi-5-SD: ~/Documents/OLLAMA/RAG-Naive — ssh mjrovai@raspi-5-SD.local — 80...
chroma_db
(ollama) mjrovai@raspi-5-SD:~/Documents/OLLAMA/RAG-Naive $ python 25-optimized_R
AG_query.py
Preloading models...
Models preloaded successfully
Loading existing vector store...

==== Optimized RAG Query System ====
Type your questions and press Enter. Type 'quit' to exit.

Your question: How is possible to automatically estimate the number of bees on a
 beehave?

Generating answer...

Question: How is possible to automatically estimate the number of bees on a beeh
ave?
Retrieving documents...
Retrieved 2 document chunks
Generating answer...
Response latency: 81.87 seconds using model: llama3.2:3b

ANSWER:
==================================================
The system uses a YOLOv8n object detection model deployed on a Raspberry Pi Zero
 2W, which enables local processing without requiring cloud transmission. This a
llows for automated bee counting at hive entrances by detecting and counting ind
ividual bees in images from the environment.
==================================================
```

25-optimized_RAG_query.py

Questions?

Prof. Marcelo J. Rovai

rovai@unifei.edu.br

UNIFEI