LLMs for me

Case Study in LLMs & Generative AI

llmsfor.me

Myles Harrison,
AI Consultant & Trainer

February 11th, 2025

NLP from scratch

Agenda

01 Introduction - Looking Back

02 A Case Study in GenAI

03 Building a GenAI-powered App

04 Conclusion - Looking Forward

Introduction - Looking Back

TABLE OF CONTENTS

A Case Study in GenAI

GenAI Case Study - from the client

> *Our healthcare network, **Devil May Care Health Clinics**, struggles with high call volumes for appointment scheduling, insurance inquiries, and general patient support, leading to long wait times and overburdened staff. We need an **AI-powered chatbot** for our website and patient portal to streamline communication, provide 24/7 assistance, and integrate with our EHR and scheduling systems while ensuring HIPAA compliance. The chatbot must **enhance patient experience**, reduce **administrative workload**, and **intelligently escalate complex issues to human staff**. Can you develop a secure, intuitive solution that meets these needs?*

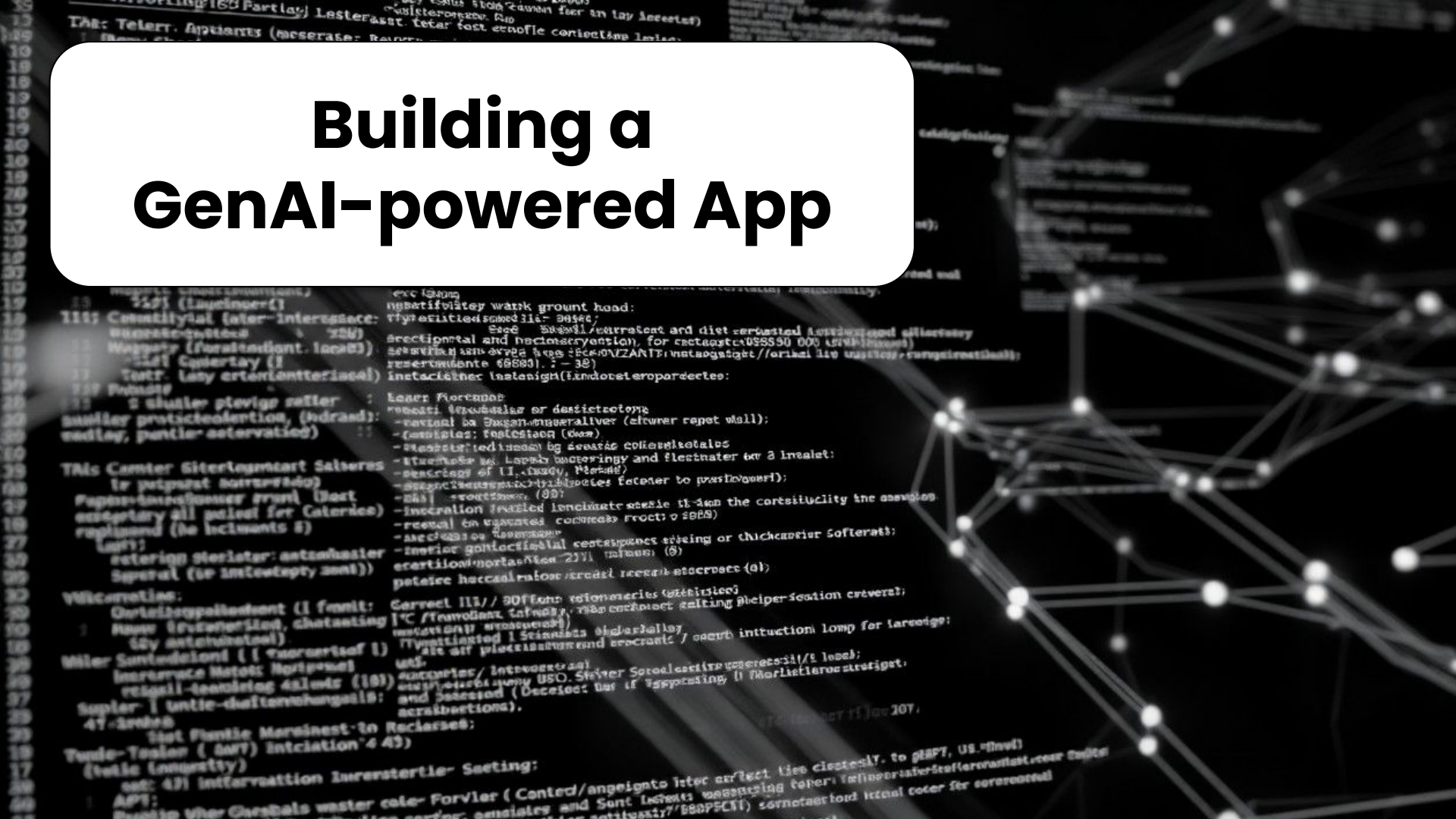
- Dr. Susan Church, Chief Operating Officer at Devil May Care Health Clinics

Requirements Gathering

This is actually the hardest part.

Don’t skip this step -
or you’ll wish you hadn’t later.

Building a GenAI-powered App



Python (Web) App Frameworks for chat

Streamlit

Chainlit

“Hello World” in Streamlit

```python
import streamlit as st

# Title of the app
st.title("Hello, World! in Streamlit")

# Displaying text
st.write("Welcome to your first Streamlit app!")
```

app.py

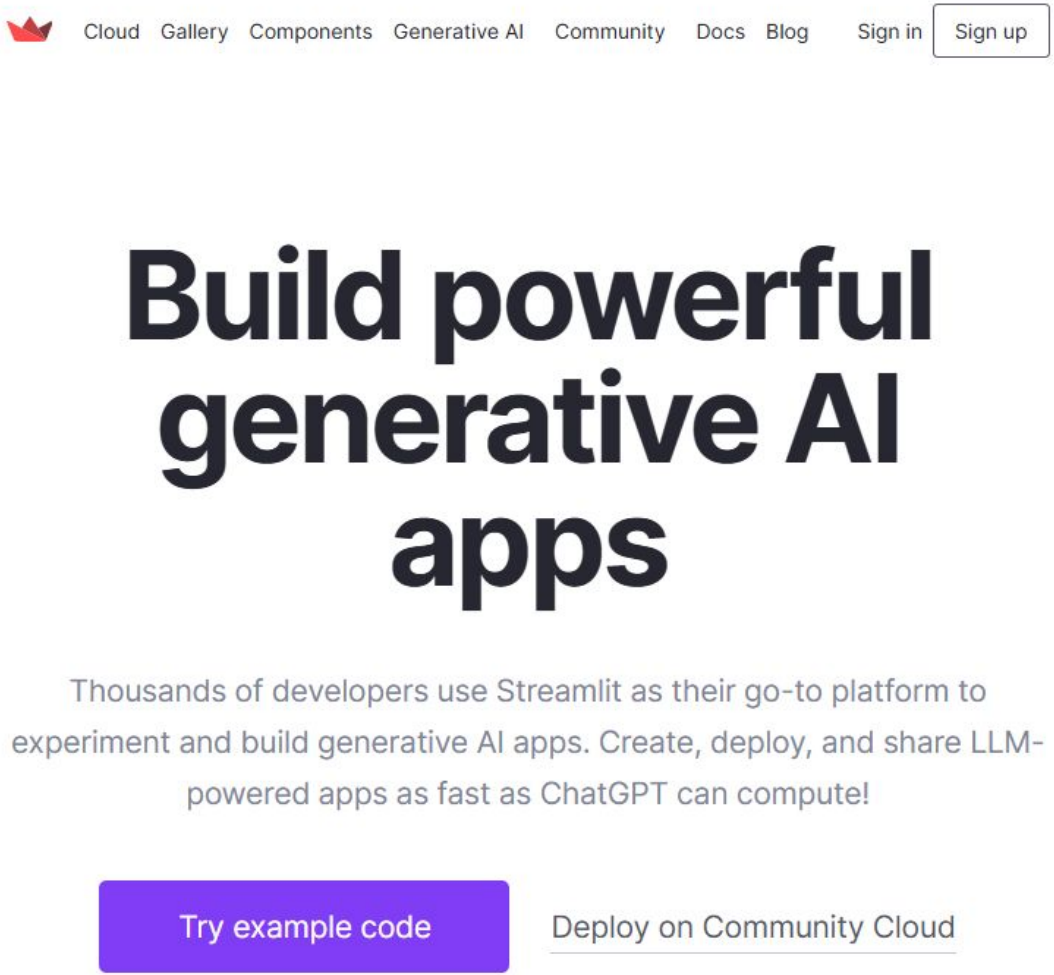
```
streamlit run app.py
```

> terminal

GenAI & LLMs in Streamlit

streamlit.io/generative-ai

docs.streamlit.io/develop/tutorials/chat-and-llm-apps/build-conversational-apps



st.chat_input

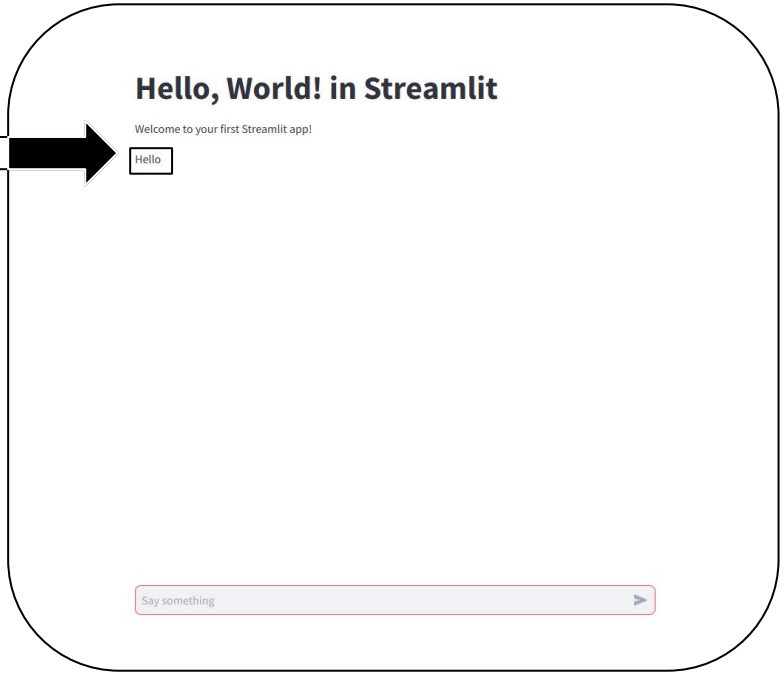
```python
import streamlit as st

# Title of the app
st.title("Hello, World! in Streamlit")

# Displaying text
st.write("Welcome to your first Streamlit app!")

# Create a chat input
st.chat_input("Say something")
```

(creates GUI element, but doesn't do anything…)

docs.streamlit.io/develop/api-reference/chat/st.chat_input

Chat elements are methods

```
...
# Create a chat input & print result to stdout
prompt = st.chat_input("Say something")
print(prompt)
st.markdown(prompt)
```



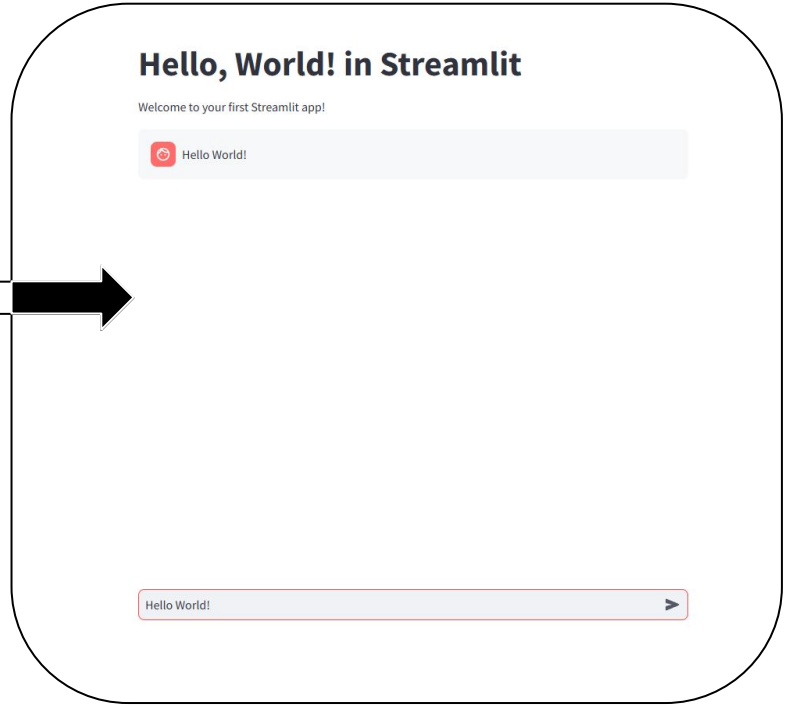
Writes output once:

- **`print()`** sent to standard out (terminal)
- **`st.markdown`** writes to streamlit app GUI

docs.streamlit.io/develop/api-reference/chat/st.chat_input

Combine with `st.chat_message`

```
...
# Create a chat input
if prompt := st.chat_input("Say something"):
    with st.chat_message("user"):
        st.markdown(prompt
```



Now we have “chat”, however this also overwrites the very first message each time.

Streamlit constantly reruns the application to change state continuously.

docs.streamlit.io/develop/api-reference/chat/st.chat_input

Add LLMs 🚀 (1 of 2)

```python
import streamlit as st
from openai import OpenAI

# Instantiate the client
client = OpenAI()

# Title of the app
st.title("Hello, World! in Streamlit")

# Displaying text
st.write("Welcome to your first Streamlit app!")

# Create a chat input
if prompt := st.chat_input("Say something"):
    with st.chat_message("user"):
        st.markdown(prompt)
```

Import OpenAI client

Instantiate client

Previous code (“user” role)

Add LLMs 🚀 (2 of 2)

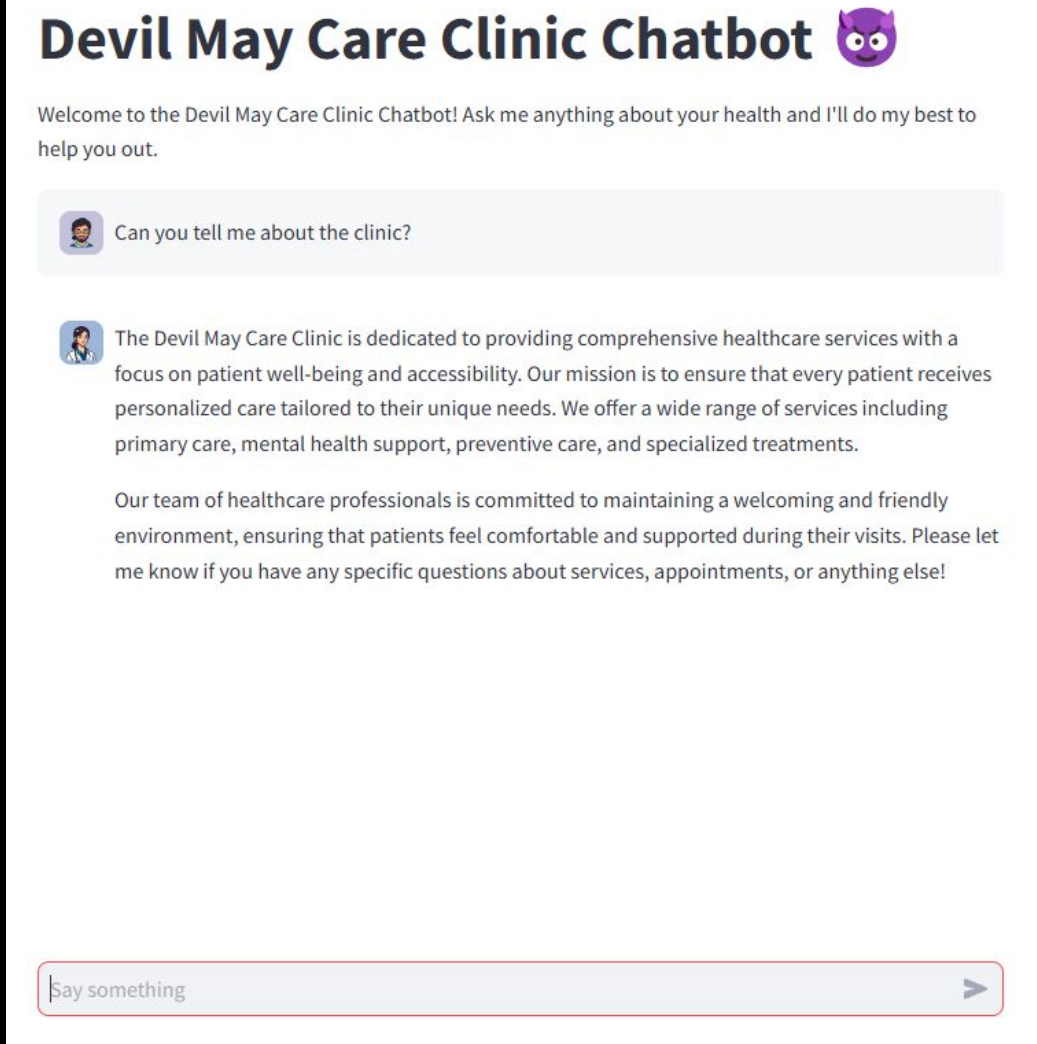
```
    # Get the response from OpenAI
    response = client.chat.completions.create(
          messages=[{"role": "user", "content":
prompt}],
          model="gpt-4o-mini"
          )
    response_text = response.choices[0].message.content

    # Write out
    with st.chat_message("assistan
        st.markdown(response_text)
```

Get response from GPT-4o-mini

“assistant” response

Customizing our app

- Add system prompt to messages array
- Changing user and assistant avatars: https://discuss.streamlit.io/t/change-default-avatars-in-st-chat/49142
- Changing CSS and styling: https://github.com/BugzTheBunny/streamlit_custom_gui



Conclusion - Looking Forward

So much to learn...

LangChain

LangGraph

DSPy *Programming—not prompting—LMs*

LlamaIndex

Weights & Biases

DeepEval.

$ the open-source LLM evaluation framework

Lumigator

Milvus

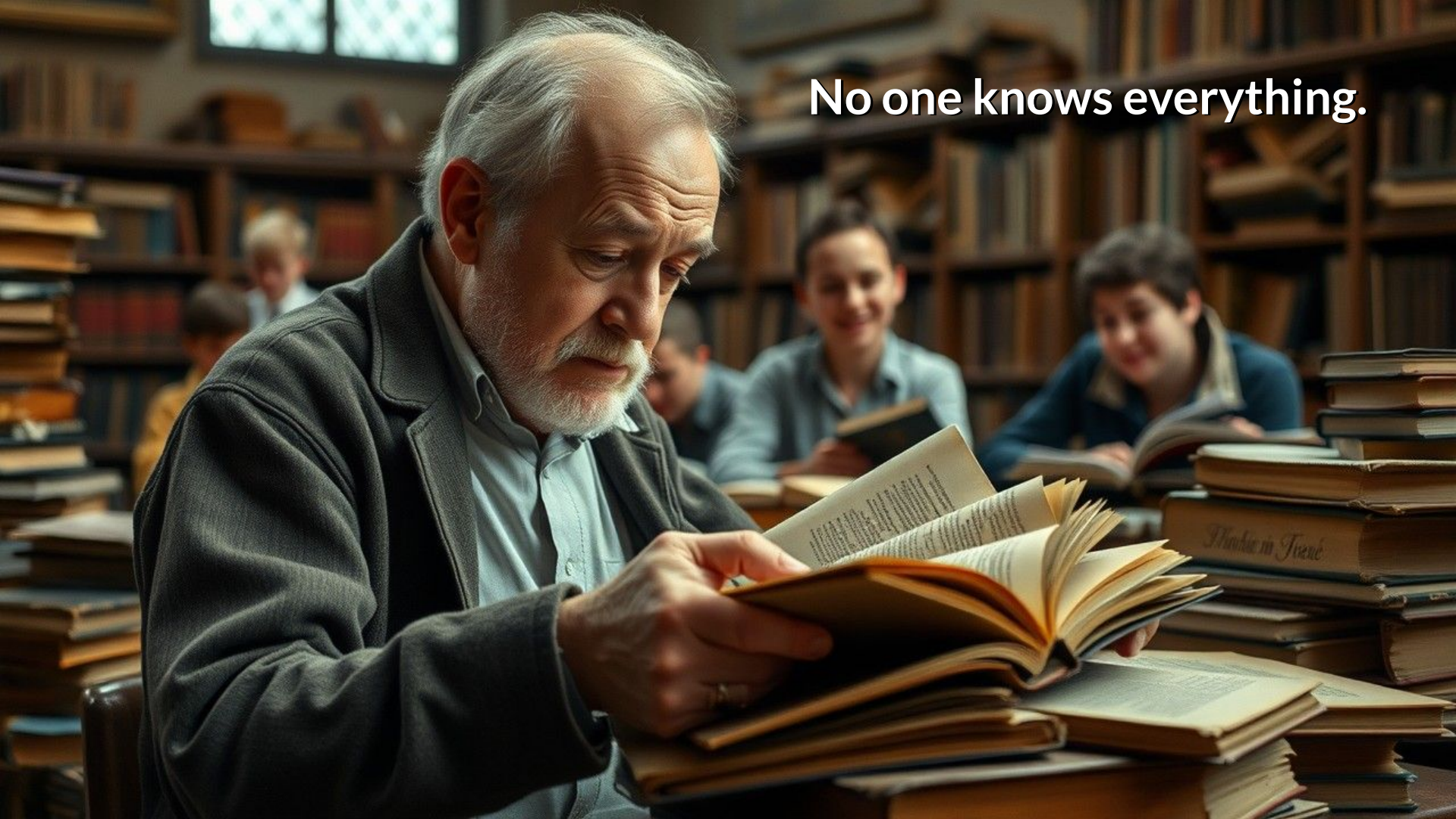
haystack
by deepset

Pinecone

Chroma

Weaviate

DuckDB


No one knows everything.

Questions?

That's it - for now 🙂

I hope you enjoyed the course. If you would like to contribute to allow me to develop more content, please consider contributing at:

nlpfromscratch.com/pwyc

I will also send out a feedback form, as the course has concluded - let me know what could be improved for future courses and what you are most interested in learning next.

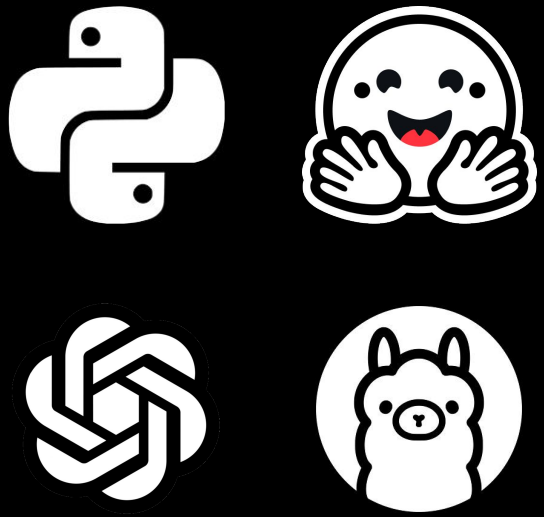
Thanks for coming!

End of Part 6

LLMsfor.me
PWYC Microcourse in LLMs and Generative AI
January 2025

Part 6 - Multimodal LLMs and Frameworks

Tuesday, February 11th, 2025



llmsfor.me

NLP from scratch