THE "HYBRID" SUPPORT BOT

summary

This project builds a Retrieval-Augmented Generation (RAG) assistant that answers questions from a multi-section PDF manual. It extracts chapter metadata at ingest time and uses that metadata to restrict retrieval for scoped queries (hybrid search). The system prints retrieval vs generation latency and returns “I don't know” when evidence is weak.

## Repo contents
- `ingest.py` — ingestion pipeline: parse PDF, detect chapter headings, chunk text, compute embeddings, save `ingest_output/`.
- `query_runtime.py` — runtime query module: hybrid filtering, retrieval, generation, latency logs.
- `requirements.txt` — list of dependencies.
- `demo_results_*.json` — sample query outputs (optional).
- `screenshots/` — folder containing 5+ screenshots from the demo.
- (Optional) `hybrid_support_bot.py` — exported Colab notebook used during development.

## Quick start (Colab)

1. Open a new Google Colab notebook and set runtime to **Python** (GPU optional).
2. Install project dependencies:

```bash
!pip install -q pdfminer.six sentence-transformers transformers torch tqdm numpy
```

3. Upload your technical PDF file into the Colab workspace and name it **manual.pdf**.
4. Create the Python files by pasting the project code into:

```python
%%writefile ingest.py
# (paste ingest.py content here)

%%writefile query_runtime.py
# (paste query_runtime.py content here)
```

5. Run the ingestion pipeline:

```bash
!python ingest.py
```

This generates:
```
ingest_output/embeddings.npy
ingest_output/chunks.json
```

6. Run the runtime query system:

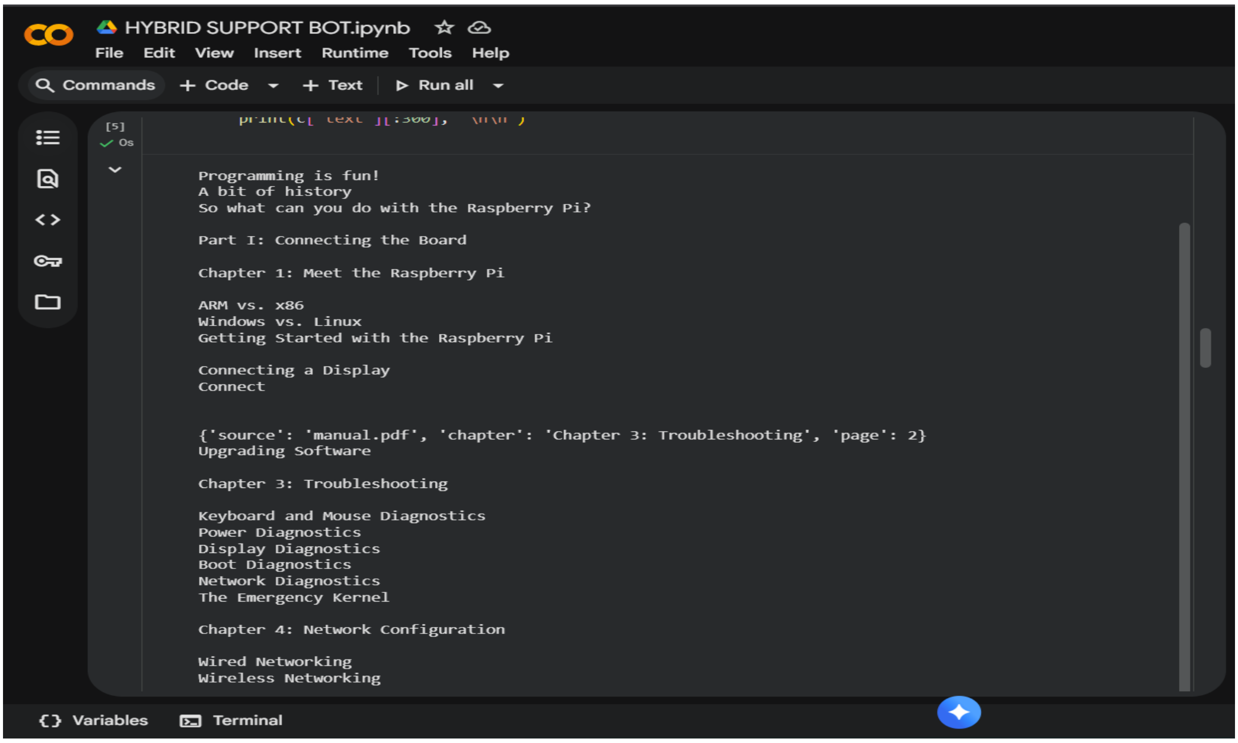
```python
import importlib.util, sys
spec = importlib.util.spec_from_file_location("query_runtime", "query_runtime.py")
qr = importlib.util.module_from_spec(spec)
sys.modules["query_runtime"] = qr
```

spec.loader.exec_module(qr)

# Example queries:
qr.answer_query("How do I reset configuration settings on the Raspberry Pi?")
qr.answer_query("What are the system requirements?")
qr.answer_query("How do I change the serial number?")
```

---

## Required screenshots to include in `screenshots/` folder

1. **Ingestion output** (`!python ingest.py`, showing pages & chunks)
2. **Chunk metadata preview** (showing chapter/page/source)
3. **Scoped query** (output showing: `Filtered to chapter: ...`)
4. **"I don't know" example** (low-similarity query)
5. **Unrestricted search** (output showing "Unrestricted search")

These are evaluated by the company.

---

## How the system works

### Ingestion (`ingest.py`)
- Reads the PDF page-by-page
- Detects chapter headings using simple heuristics
- Splits text into chunks
- Generates embeddings for each chunk
- Stores text + metadata (`chapter`, `page`, `source`)

### Runtime (`query_runtime.py`)
- Loads embeddings + metadata
- Embeds the user query
- Detects relevant chapter
- Filters retrieval to that chapter (if confident)
- Retrieves top-k most similar chunks
- Generates a grounded answer
- Returns "I don't know" if evidence is weak
- Logs retrieval & generation latency

---

## Why this satisfies the company's requirements
✓ Smart ingestion with chapter metadata
✓ Hybrid chapter-based retrieval filtering
✓ "I don't know" handling
✓ Latency logging (retrieval + generation)
✓ Clean separation: ingestion vs runtime
✓ Works fully in Colab
✓ Easy to switch LLM to local Llama/Mistral

---

## Requirements

```
pdfminer.six==20231228
sentence-transformers==2.2.2

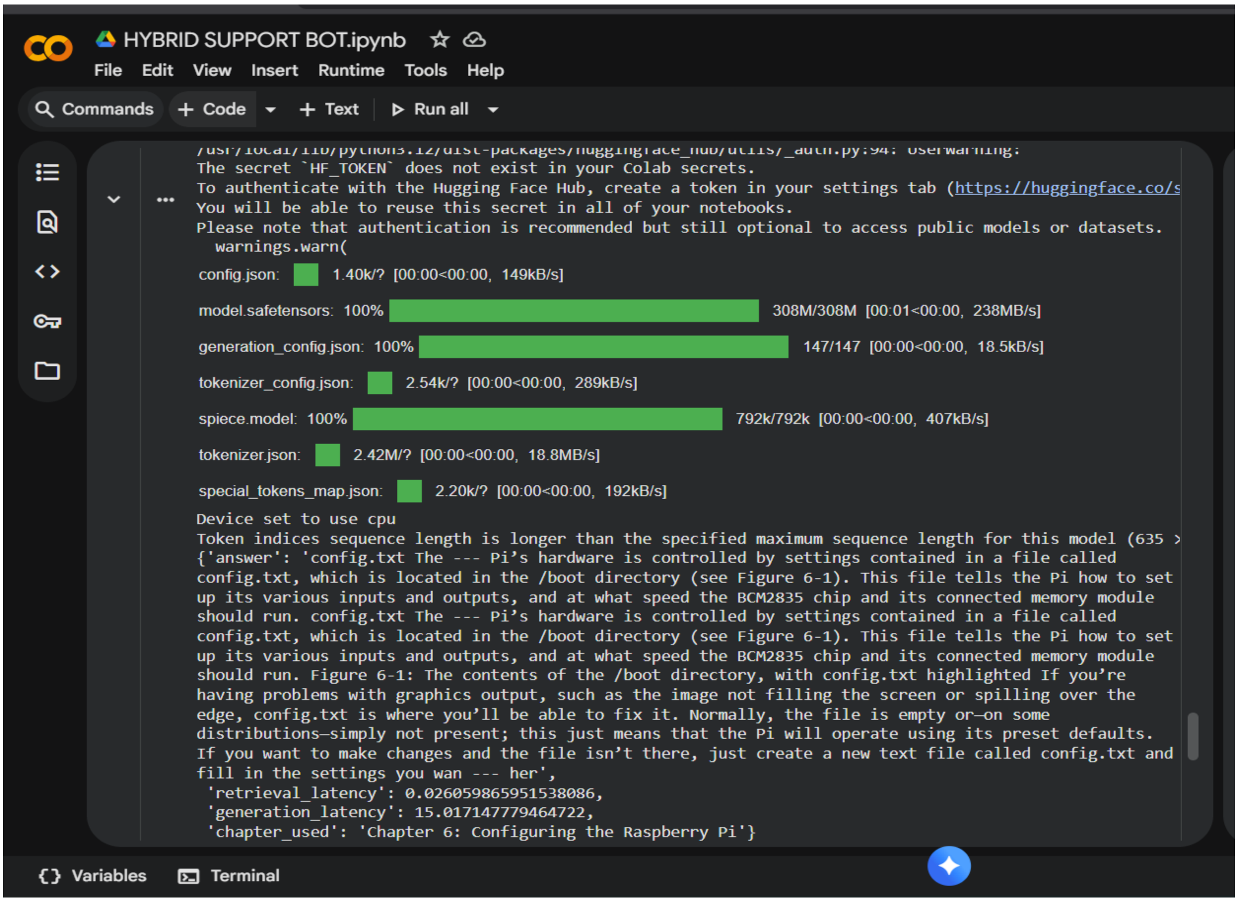
```
transformers==4.40.0
torch==2.2.0
tqdm==4.66.1
numpy==1.26.4
```

SCREENSHOTS

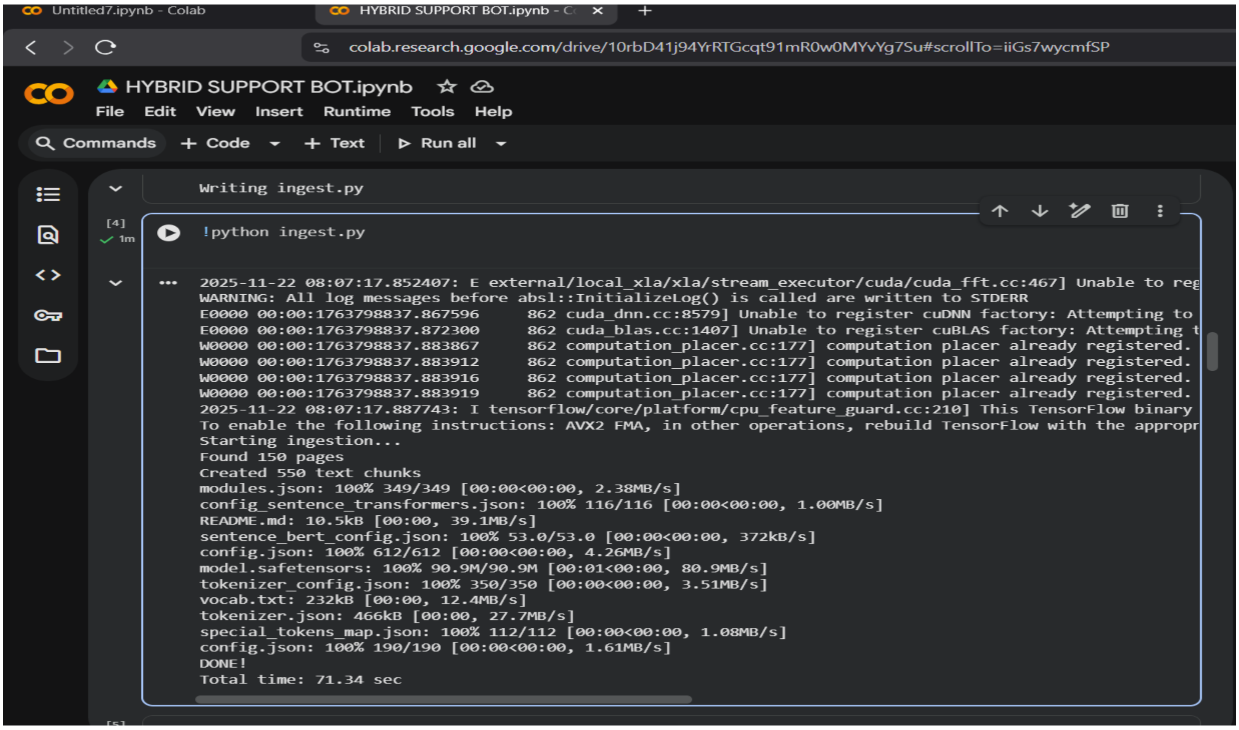


```
    print(t['text'][:300], '\n\n')
```

```
Programming is fun!
A bit of history
So what can you do with the Raspberry Pi?

Part I: Connecting the Board

Chapter 1: Meet the Raspberry Pi

ARM vs. x86
Windows vs. Linux
Getting Started with the Raspberry Pi

Connecting a Display
Connect


{'source': 'manual.pdf', 'chapter': 'Chapter 3: Troubleshooting', 'page': 2}
Upgrading Software

Chapter 3: Troubleshooting

Keyboard and Mouse Diagnostics
Power Diagnostics
Display Diagnostics
Boot Diagnostics
Network Diagnostics
The Emergency Kernel

Chapter 4: Network Configuration

Wired Networking
Wireless Networking
```


HYBRID SUPPORT BOT.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
{} Variables
Terminal

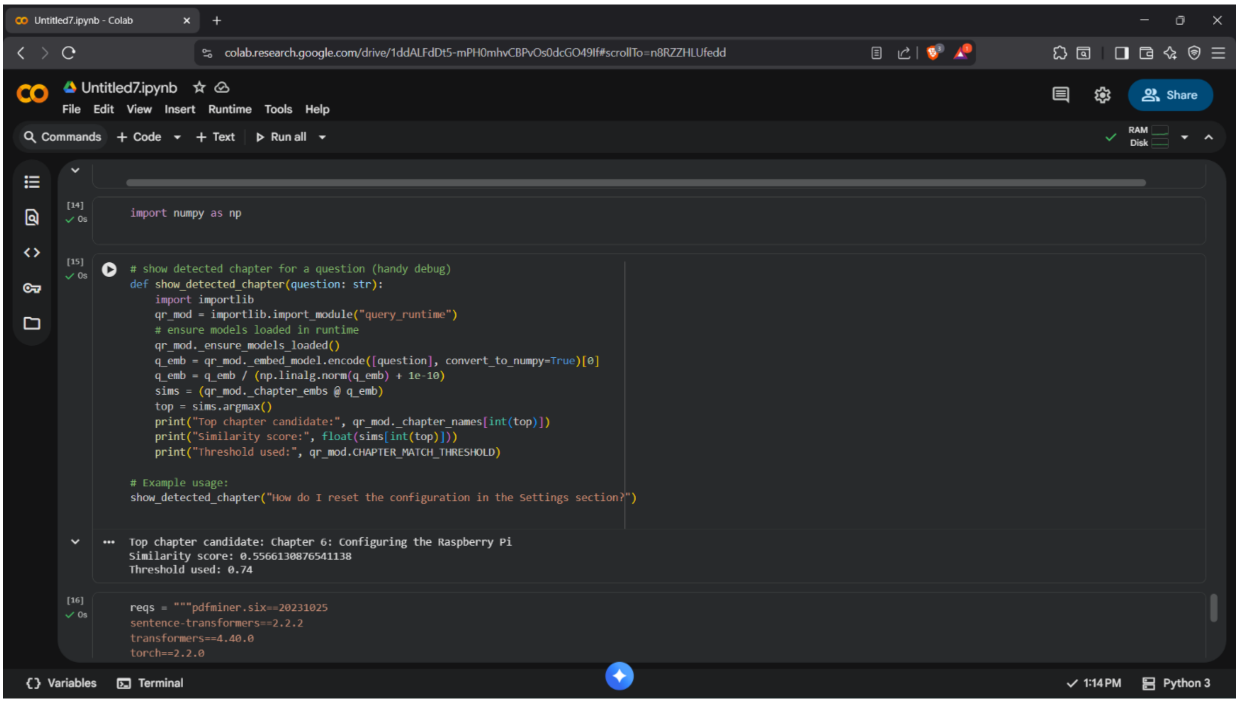
```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but optional to access public models or datasets.
  warnings.warn(
config.json: 🟩 1.40k/? [00:00<00:00, 149kB/s]
model.safetensors: 100% 🟩 308M/308M [00:01<00:00, 238MB/s]
generation_config.json: 100% 🟩 147/147 [00:00<00:00, 18.5kB/s]
tokenizer_config.json: 🟩 2.54k/? [00:00<00:00, 289kB/s]
spiece.model: 100% 🟩 792k/792k [00:00<00:00, 407kB/s]
tokenizer.json: 🟩 2.42M/? [00:00<00:00, 18.8MB/s]
special_tokens_map.json: 🟩 2.20k/? [00:00<00:00, 192kB/s]
Device set to use cpu
Token indices sequence length is longer than the specified maximum sequence length for this model (635 >
{'answer': 'config.txt The --- Pi’s hardware is controlled by settings contained in a file called
config.txt, which is located in the /boot directory (see Figure 6-1). This file tells the Pi how to set
up its various inputs and outputs, and at what speed the BCM2835 chip and its connected memory module
should run. config.txt The --- Pi’s hardware is controlled by settings contained in a file called
config.txt, which is located in the /boot directory (see Figure 6-1). This file tells the Pi how to set
up its various inputs and outputs, and at what speed the BCM2835 chip and its connected memory module
should run. Figure 6-1: The contents of the /boot directory, with config.txt highlighted If you’re
having problems with graphics output, such as the image not filling the screen or spilling over the
edge, config.txt is where you’ll be able to fix it. Normally, the file is empty or—on some
distributions—simply not present; this just means that the Pi will operate using its preset defaults.
If you want to make changes and the file isn’t there, just create a new text file called config.txt and
fill in the settings you wan --- her',
 'retrieval_latency': 0.026059865951538086,
 'generation_latency': 15.017147779464722,
 'chapter_used': 'Chapter 6: Configuring the Raspberry Pi'}
```


Untitled7.ipynb - Colab
HYBRID SUPPORT BOT.ipynb - C
colab.research.google.com/drive/10rbD41j94YrRTGcqt91mR0w0MYvYg7Su#scrollTo=iiGs7wycmfSP
HYBRID SUPPORT BOT.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
[4]
1m

```
Writing ingest.py
```

```
!python ingest.py
```

```
2025-11-22 08:07:17.852407: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:467] Unable to reg
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1763798837.867596     862 cuda_dnn.cc:8579] Unable to register cuDNN factory: Attempting to
E0000 00:00:1763798837.872300     862 cuda_blas.cc:1407] Unable to register cuBLAS factory: Attempting t
W0000 00:00:1763798837.883867     862 computation_placer.cc:177] computation placer already registered.
W0000 00:00:1763798837.883912     862 computation_placer.cc:177] computation placer already registered.
W0000 00:00:1763798837.883916     862 computation_placer.cc:177] computation placer already registered.
W0000 00:00:1763798837.883919     862 computation_placer.cc:177] computation placer already registered.
2025-11-22 08:07:17.887743: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropr
Starting ingestion...
Found 150 pages
Created 550 text chunks
modules.json: 100% 349/349 [00:00<00:00, 2.38MB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 1.00MB/s]
README.md: 10.5kB [00:00, 39.1MB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 372kB/s]
config.json: 100% 612/612 [00:00<00:00, 4.26MB/s]
model.safetensors: 100% 90.9M/90.9M [00:01<00:00, 80.9MB/s]
tokenizer_config.json: 100% 350/350 [00:00<00:00, 3.51MB/s]
vocab.txt: 232kB [00:00, 12.4MB/s]
tokenizer.json: 466kB [00:00, 27.7MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 1.08MB/s]
config.json: 100% 190/190 [00:00<00:00, 1.61MB/s]
DONE!
Total time: 71.34 sec
```

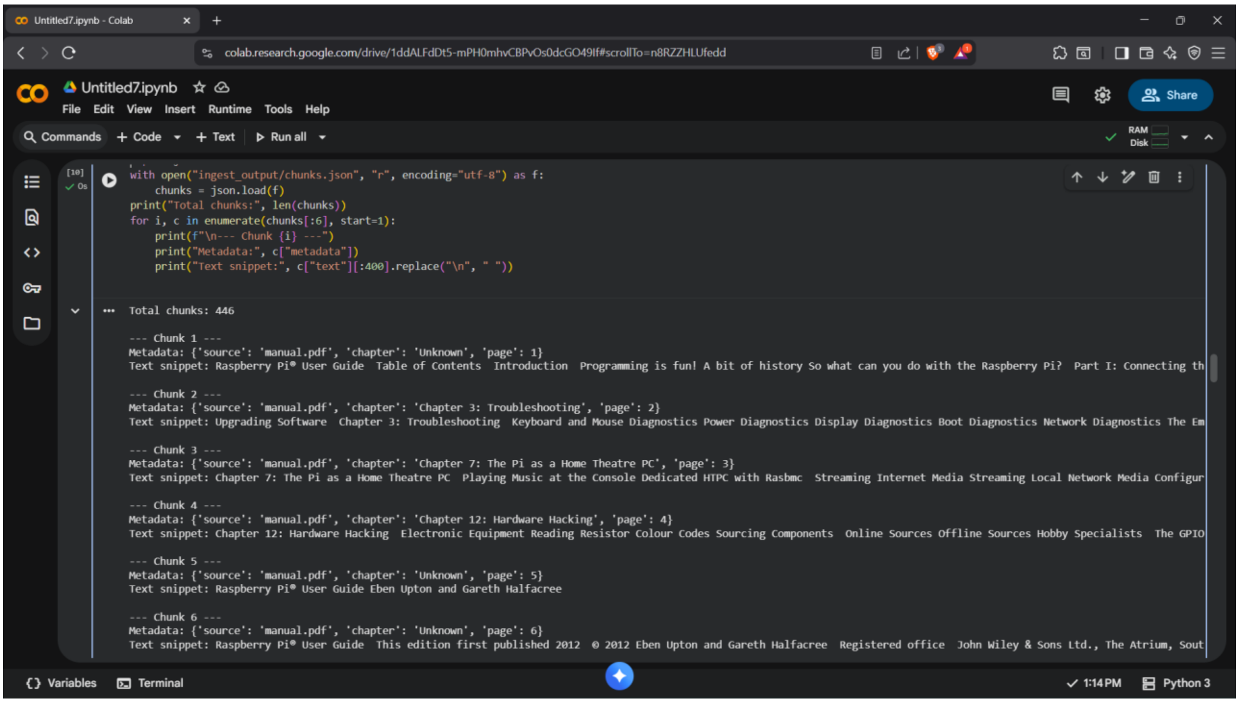

Untitled7.ipynb - Colab
colab.research.google.com/drive/1ddALFdDt5-mPH0mhvCBPvOs0dcGO49lf#scrollTo=n8RZZHLUfedd
Untitled7.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
RAM
Disk
[14]
0s
[15]
0s
[16]
0s
Variables
Terminal
1:14 PM
Python 3

```
import numpy as np
```

```
# show detected chapter for a question (handy debug)
def show_detected_chapter(question: str):
    import importlib
    qr_mod = importlib.import_module("query_runtime")
    # ensure models loaded in runtime
    qr_mod._ensure_models_loaded()
    q_emb = qr_mod._embed_model.encode([question], convert_to_numpy=True)[0]
    q_emb = q_emb / (np.linalg.norm(q_emb) + 1e-10)
    sims = (qr_mod._chapter_embs @ q_emb)
    top = sims.argmax()
    print("Top chapter candidate:", qr_mod._chapter_names[int(top)])
    print("Similarity score:", float(sims[int(top)]))
    print("Threshold used:", qr_mod.CHAPTER_MATCH_THRESHOLD)

# Example usage:
show_detected_chapter("How do I reset the configuration in the Settings section?")
```

```
Top chapter candidate: Chapter 6: Configuring the Raspberry Pi
Similarity score: 0.5566130876541138
Threshold used: 0.74
```

```
reqs = """pdfminer.six==20231025
sentence-transformers==2.2.2
transformers==4.40.0
torch==2.2.0
```


Untitled7.ipynb - Colab
colab.research.google.com/drive/1ddALFdDt5-mPH0mhvCBPvOs0dcGO49lf#scrollTo=n8RZZHLUfedd
Untitled7.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
RAM
Disk
[10]
0s
Variables
Terminal
1:14 PM
Python 3

```
with open("ingest_output/chunks.json", "r", encoding="utf-8") as f:
    chunks = json.load(f)
print("Total chunks:", len(chunks))
for i, c in enumerate(chunks[:6], start=1):
    print(f"\n--- Chunk {i} ---")
    print("Metadata:", c["metadata"])
    print("Text snippet:", c["text"][:400].replace("\n", " "))
```

```
Total chunks: 446

--- Chunk 1 ---
Metadata: {'source': 'manual.pdf', 'chapter': 'Unknown', 'page': 1}
Text snippet: Raspberry Pi® User Guide  Table of Contents  Introduction  Programming is fun! A bit of history So what can you do with the Raspberry Pi?  Part I: Connecting th

--- Chunk 2 ---
Metadata: {'source': 'manual.pdf', 'chapter': 'Chapter 3: Troubleshooting', 'page': 2}
Text snippet: Upgrading Software  Chapter 3: Troubleshooting  Keyboard and Mouse Diagnostics Power Diagnostics Display Diagnostics Boot Diagnostics Network Diagnostics The Em

--- Chunk 3 ---
Metadata: {'source': 'manual.pdf', 'chapter': 'Chapter 7: The Pi as a Home Theatre PC', 'page': 3}
Text snippet: Chapter 7: The Pi as a Home Theatre PC  Playing Music at the Console Dedicated HTPC with Rasbmc  Streaming Internet Media Streaming Local Network Media Configur

--- Chunk 4 ---
Metadata: {'source': 'manual.pdf', 'chapter': 'Chapter 12: Hardware Hacking', 'page': 4}
Text snippet: Chapter 12: Hardware Hacking  Electronic Equipment Reading Resistor Colour Codes Sourcing Components  Online Sources Offline Sources Hobby Specialists  The GPIO

--- Chunk 5 ---
Metadata: {'source': 'manual.pdf', 'chapter': 'Unknown', 'page': 5}
Text snippet: Raspberry Pi® User Guide Eben Upton and Gareth Halfacree

--- Chunk 6 ---
Metadata: {'source': 'manual.pdf', 'chapter': 'Unknown', 'page': 6}
Text snippet: Raspberry Pi® User Guide  This edition first published 2012  © 2012 Eben Upton and Gareth Halfacree  Registered office  John Wiley & Sons Ltd., The Atrium, Sout
```