

Finally a Local RAG That WORKS!! (+ FULL RAG Pipeline)

Thomas Janssen
11.2K subscribers

Subscribe

Like 693 | Dislike

32,081 views May 27, 2025 #ollama #python #retrievalaugmentedgeneration

Build a local RAG (Retrieval Augmented Generation) Chatbot with Python, LangChain, Ollama, ChromaDB and Streamlit!

Get your \$15 free Bright Data credits ↗
<https://brdta.com/tomstechacademy>

timestamps ↗

00:00 - Intro
01:23 - What is Retrieval Augmented Generation and why is Agentic RAG better?
04:08 - How to install Ollama?
04:39 - How to install models on Ollama?
06:37 - What are embeddings?
08:19 - How to run embedding models local with Ollama?
09:00 - Building a RAG Pipeline
10:17 - How to download and install the Python script?
11:18 - Example of building a RAG pipeline
12:52 - Configuring the script (.env file)
14:02 - Scraping data for our RAG Pipeline
15:40 - Getting a Bright Data API Key
20:50 - Executing your RAG Chatbot with Streamlit

github.com/ThomasJanssen-tech/Local-RAG-with-Ollama

ThomasJanssen-tech / Local-RAG-with-Ollama

Type / to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Local-RAG-with-Ollama Public

main 1 Branch 0 Tags

Go to file Add file Code

ThomasJanssen-tech updated readme c145b33 · 25 minutes ago 5 Commits

File	Changed To	Message
.env.example	llama3:2.3b	yesterday
.gitignore	minor tweaks	yesterday
1_scraping_wikipedia.py	minor tweaks	yesterday
2_chunking_embedding_ingestion.py	minor tweaks	yesterday
3_chatbot.py	llama3:2.3b	yesterday
example_chunking.py	first commit	yesterday
example_embedding.py	first commit	yesterday
example_retriever.py	first commit	yesterday
keywords.xlsx	minor tweaks	yesterday
readme.md	updated readme	25 minutes ago
requirements.txt	first commit	yesterday
thumbnail_small.png	updated thumbnail	yesterday

About

Build a 100% local Retrieval Augmented Generation (RAG) system with Python, LangChain, Ollama and ChromaDB!

thomasjanssen.tech/

python langchain langchain-python chromadb retrieval-augmented-generation ollama

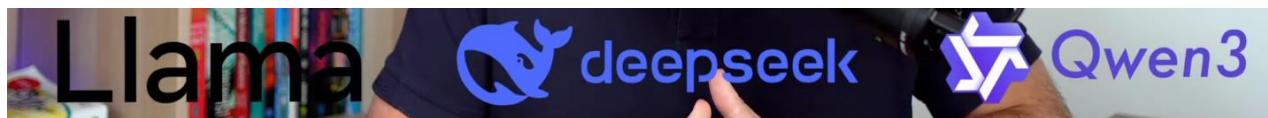
Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published



```

C:\WINDOWS\system32\cmd. X + ^
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Thomas TA.THOM_DESK_22>ollama list
NAME           ID      SIZE    MODIFIED
mxbai-embed-large:latest 468836162de7 669 MB  3 days ago
qwen3:8b        e4b5fd7f8af0 5.2 GB   4 days ago
nomic-embed-text:latest 0a109f422b47 274 MB   4 weeks ago
llama3.2:1b     baf6a787fdff 1.3 GB   2 months ago
llama3.2:3b     a80c4f17acd5 2.0 GB   3 months ago
llama3.2:latest a80c4f17acd5 2.0 GB   6 months ago

C:\Users\Thomas TA.THOM_DESK_22>ollama3.2:3b
>>> when is LangChain created and by whom?
I don't know any information about a project or entity called "LangChain". It's possible that it's a relatively new or lesser-known project, or it may not exist at all.

however, I did find information on LangChain, which is an open-source blockchain development framework for Solidity. LangChain was created in 2021 by Alex Attanasiou and other contributors to the Ethereum community.

LangChain aims to simplify and streamline smart contract development on the Ethereum blockchain by providing a set of tools and libraries that make it easier to build and deploy complex blockchain applications.

>>> Send a message (/? for help)

```

This is wrong, RAG can help

Agentic RAG Chatbot

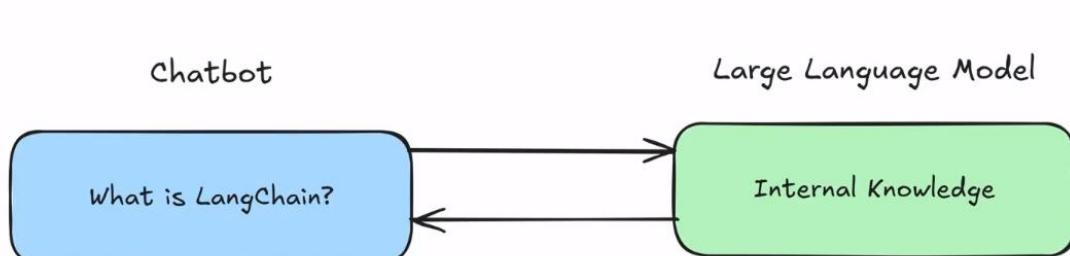
 when is langchain created and by whom?

 LangChain was launched in October 2022 as an open source project by Harrison Chase, while working at machine learning startup Robust Intelligence. The project quickly garnered popularity, with improvements from hundreds of contributors on GitHub, trending discussions on Twitter, lively activity on the project's Discord server, many YouTube tutorials, and meetups in San Francisco and London.

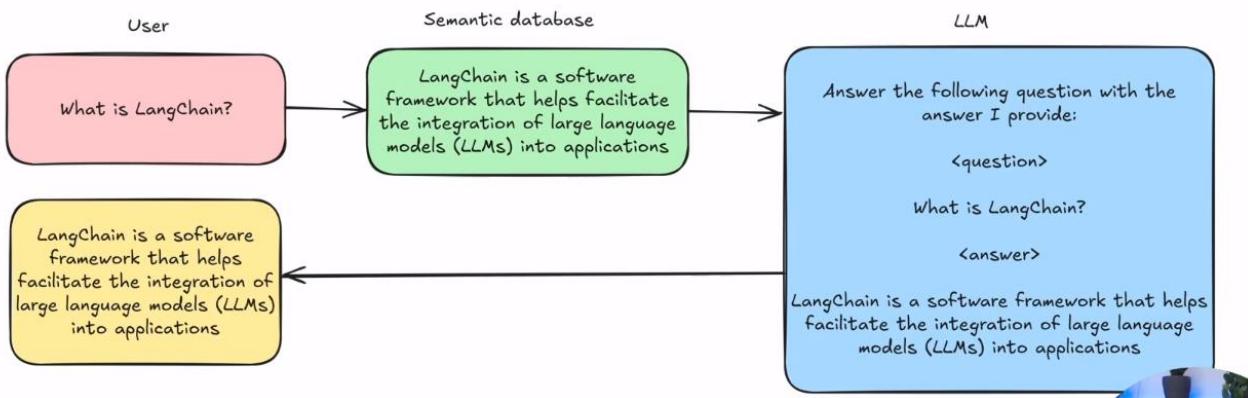
Source: <https://en.wikipedia.org/wiki/LangChain>



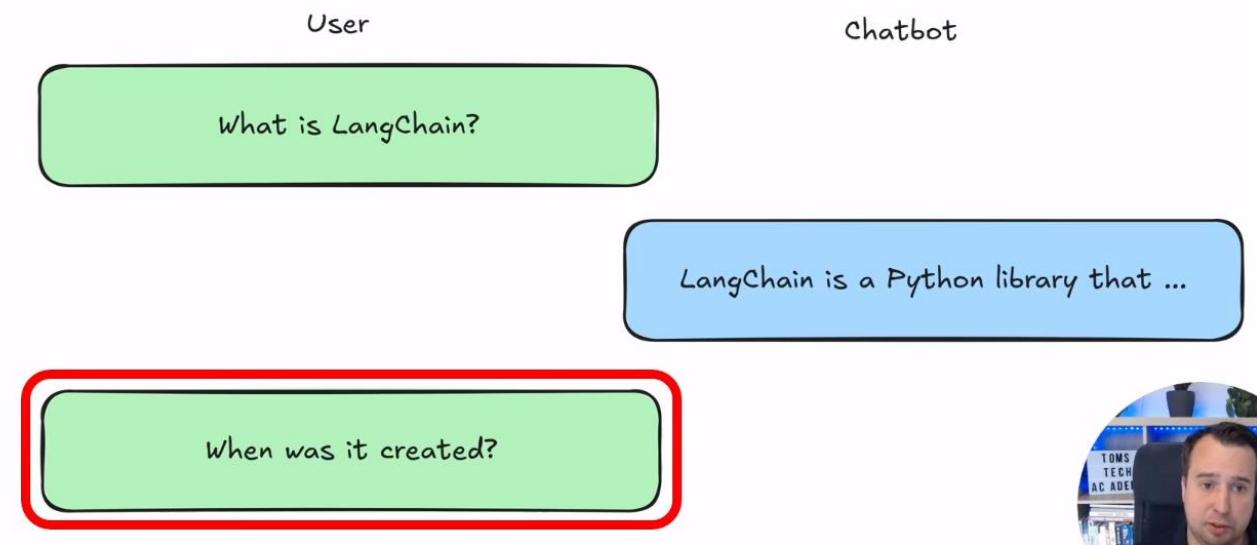
Chatbot without RAG



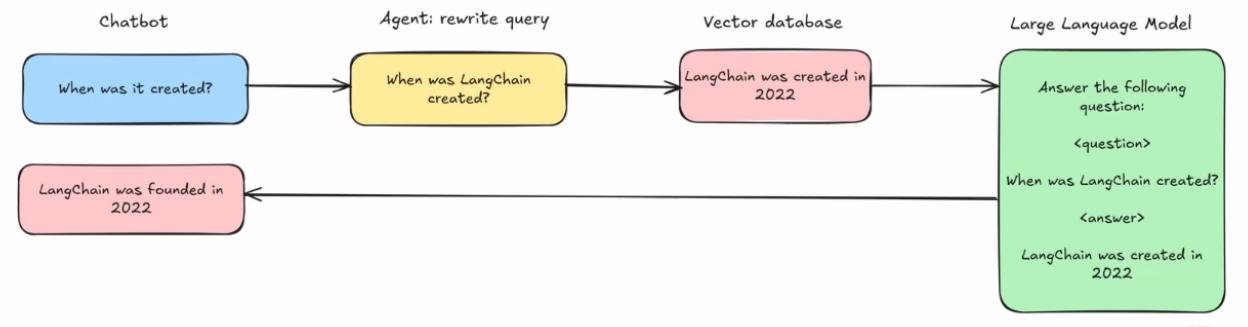
Chatbot with RAG



Chat conversation



Agentic RAG



```
C:\Users\Thomas TA.THOM_DESK_22>ollama list
NAME           ID      SIZE    MODIFIED
qwen3:8b       e4b5fd7f8af0  5.2 GB  23 hours ago
mxbai-embed-large:latest  468836162de7  669 MB  2 weeks ago
nomic-embed-text:latest  0a109f422b47  274 MB  3 weeks ago
llama3.2:1b     baf6a787fdff  1.3 GB  8 weeks ago
llama3.2:3b     a80c4f17acd5  2.0 GB  3 months ago
llama3.2:latest  a80c4f17acd5  2.0 GB  6 months ago
```

C:\Users\Thomas TA.THOM_DESK_22>

We need a chat model, an embedding model



llama3.2:3b

17.4M Downloads Updated 7 months ago

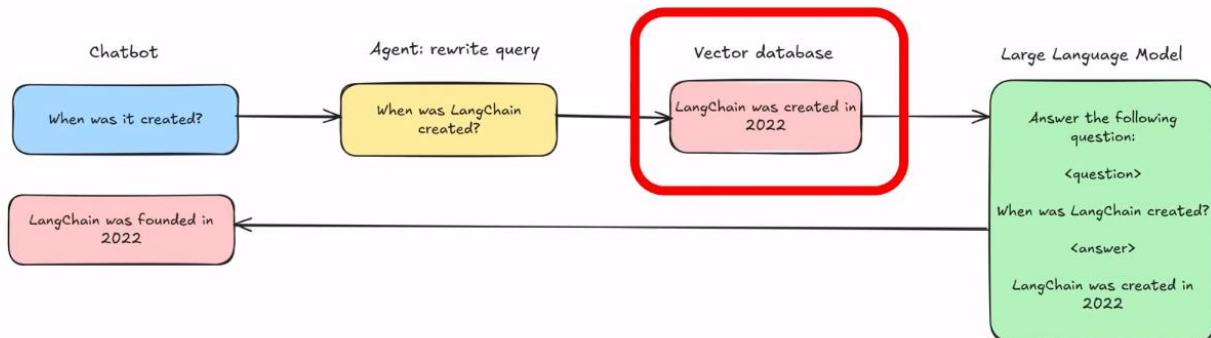
Meta's Llama 3.2 goes small with 1B and 3B models.

tools 1b 3b

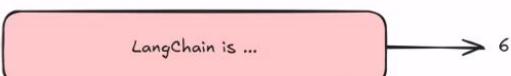
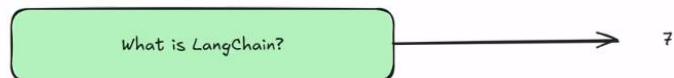
Updated 7 months ago a80c4f17acd5 · 2.0GB

model arch: llama · parameters: 3.21B · quantization: Q4_K_M 2.0GB

Agentic RAG



Embeddings explained



Embeddings explained

"Ollama is the easiest way to get up and running with large language models."



```
[-0.15521588921546936,  
 -0.3130679428577423,  
 -0.2622824013233185,  
 -0.10730823874473572,  
 ...  
 0.26006409525871277,  
 0.14494779706001282,  
 -0.01514953002333641,  
 0.04403747618198395]
```

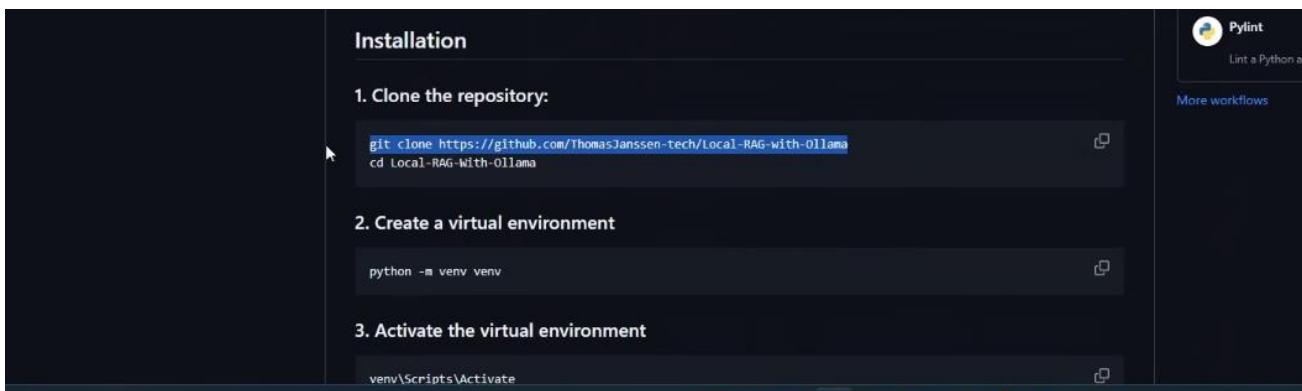
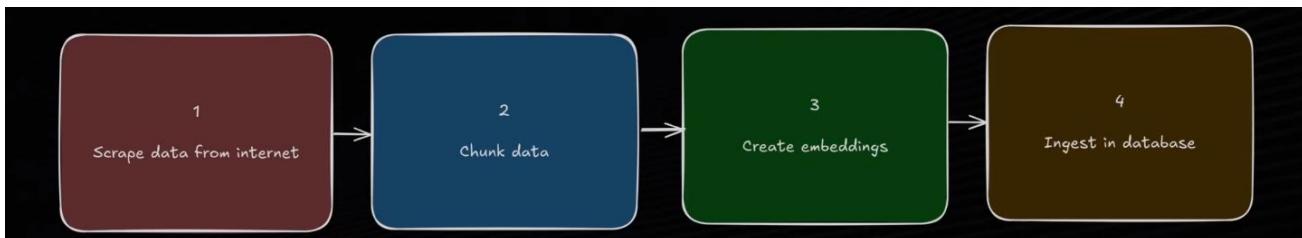


mxbai-embed-large

3.2M Downloads Updated 1 year ago
State-of-the-art large embedding model from mixedbread.ai
embedding 335m

Models

[View all →](#)



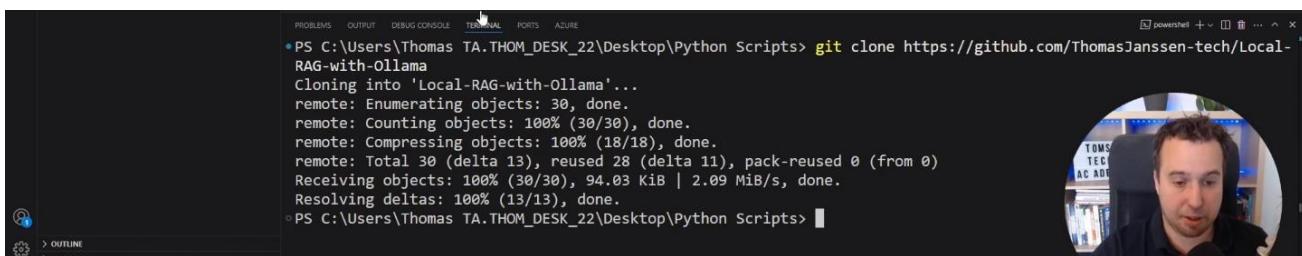
Installation

1. Clone the repository:

```
git clone https://github.com/ThomasJanssen-tech/Local-RAG-with-Ollama  
cd Local-RAG-With-Ollama
```
2. Create a virtual environment:

```
python -m venv venv
```
3. Activate the virtual environment:

```
venv\Scripts\Activate
```



```
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts> git clone https://github.com/ThomasJanssen-tech/Local-RAG-with-Ollama  
Cloning into 'Local-RAG-with-Ollama'...  
remote: Enumerating objects: 30, done.  
remote: Counting objects: 100% (30/30), done.  
remote: Compressing objects: 100% (18/18), done.  
remote: Total 30 (delta 13), reused 28 (delta 11), pack-reused 0 (from 0)  
Receiving objects: 100% (30/30), 94.03 KiB | 2.09 MiB/s, done.  
Resolving deltas: 100% (13/13), done.  
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts>
```

```
File Edit Selection View Go Run Terminal Help  
EXPLORER  
LOCAL-RAG-WITH-Ollama  
|.env.example  
|.gitignore  
1_scraping_wikipedia.py  
2_chunking_embedding_ ingestion.py  
3_chatbot.py  
example_chunking.py  
example_embedding.py  
example_retriever.py  
keywords.xlsx  
readme.md  
requirements.txt  
thumbnail_small.png  
requirements.txt  
1 altair==5.5.0  
2 annotated-types==0.7.0  
3 anyio==4.9.0  
4 asgiref==3.8.1  
5 attrs==25.3.0  
6 backoff==2.2.1  
7 bcrypt==4.3.0  
8 blinker==1.9.0  
9 build==1.2.2.post1  
10 cachetools==5.5.2  
11 certifi==2025.4.26  
12 charset-normalizer==3.4.2  
13 chromadb==1.0.9  
14 click==8.1.8  
15 colorama==0.4.6  
16 coloredlogs==15.0.1
```

```
File Edit Selection View Go Run Terminal Help  
EXPLORER  
LOCAL-RAG-WITH-Ollama  
|.env  
|.env.example  
|.gitignore  
1_scraping_wikipedia.py  
2_chunking_embedding_ ingestion.py  
3_chatbot.py  
example_chunking.py  
example_embedding.py  
example_retriever.py  
keywords.xlsx  
readme.md  
requirements.txt  
thumbnail_small.png  
1_scraping_wikipedia.py  
1 ##### 1. IMPORTING MODULES AND INITIALIZING VARIABLES #####  
2 #####  
3 #####  
4  
5 from dotenv import load_dotenv  
6 import os  
7 import requests  
8 import json  
9 import pandas as pd  
10 import glob  
11  
12 pd.options.mode.chained_assignment = None  
13  
14 load_dotenv()  
15  
16 ##### HEADERS (DON'T CHANGE) #####  
17 headers = {  
18     'Authorization': 'Bearer '+os.getenv('BRIGHTDATA_API_KEY'),  
19     'Content-Type': 'application/json',  
20 }  
21  
22 headers_status = {  
23     'Authorization': 'Bearer '+os.getenv('BRIGHTDATA_API_KEY'),  
24 }  
25  
26 keywords = pd.read_excel("keywords.xlsx")  
27  
28 ##### 2. TF Snapshot ID IS NOT SET IN .XLSX FILE. TRIG.  
29 #####  
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> python -m venv venv  
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> venv\Scripts\Activate  
% venv% PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> pip install -r requirements.txt
```

```
File Edit Selection View Go Run Terminal Help  
EXPLORER  
LOCAL-RAG-WITH-Ollama  
|.env.example  
|.gitignore  
1_scraping_wikipedia.py  
2_chunking_embedding_ ingestion.py  
3_chatbot.py  
example_chunking.py  
example_embedding.py  
example_retriever.py  
keywords.xlsx  
readme.md  
requirements.txt  
thumbnail_small.png  
1_scraping_wikipedia.py  
1 ##### 1. IMPORTING MODULES AND INITIALIZING VARIABLES #####  
2 #####  
3 #####  
4  
5 from dotenv import load_dotenv  
6 import os  
7 import requests  
8 import json  
9 import pandas as pd  
10 import glob  
11  
12 pd.options.mode.chained_assignment = None  
13  
14 load_dotenv()  
15  
16 ##### HEADERS (DON'T CHANGE) #####  
17 headers = {  
18     'Authorization': 'Bearer '+os.getenv('BRIGHTDATA_API_KEY'),  
19     'Content-Type': 'application/json',  
20 }  
21  
22 headers_status = {  
23     'Authorization': 'Bearer '+os.getenv('BRIGHTDATA_API_KEY'),  
24 }  
25  
26 keywords = pd.read_excel("keywords.xlsx")  
27  
28 ##### 2. TF Snapshot ID IS NOT SET IN .XLSX FILE. TRIG.  
29 #####  
TOM  
TEC  
AC AD
```



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

```

example_chunking.py
example_chunking.py > ...
1 from langchain_text_splitters import RecursiveCharacterTextSplitter
2
3
4 # source: https://python.langchain.com/docs/introduction/
5 entire_text = """
6
7 LangChain is a framework for developing applications powered by large language models (LLMs).
8
9 LangChain simplifies every stage of the LLM application lifecycle:
10
11 Development: Build your applications using LangChain's open-source components and third-party inte
12 Productionization: Use LangSmith to inspect, monitor and evaluate your applications, so that you c
13 Deployment: Turn your LangGraph applications into production-ready APIs and Assistants with LangGr
14 Diagram outlining the hierarchical organization of the LangChain framework, displaying the interco
15 LangChain implements a standard interface for large language models and related technologies, such
16
17 Select chat model:
18 pip install -qU "langchain[openai]"
19
20 import getpass
21 import os
22
23 if not os.environ.get("OPENAI_API_KEY"):
24     os.environ["OPENAI_API_KEY"] = getpass.getpass("Enter API key for OpenAI: ")
25
26 from langchain.chat_models import init_chat_model
27
28 model = init_chat_model("gpt-4o-mini", model_provider="openai")
29

```

OUTLINE TIMELINE



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

```

example_chunking.py
example_chunking.py > ...
58 Check out LangGraph-specific how-tos here.
59
60 Conceptual guide
61 Introductions to all the key parts of LangChain you'll need to know! Here you'll find high level e
62
63 For a deeper dive into LangGraph concepts, check out this page.
64
65 """
66
67
68 text_splitter = RecursiveCharacterTextSplitter(
69     # Set a really small chunk size, just to show.
70     chunk_size=100,
71     chunk_overlap=20,
72     length_function=len,
73     is_separator_regex=False,
74 )
75
76 texts = text_splitter.create_documents([entire_text])
77
78 print(f"Total chunks: {len(texts)}")
79
80 print(f"First chunk: {texts[0].page_content}")

```

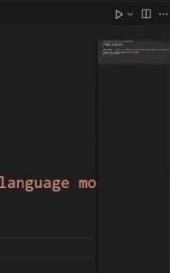


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/example_chunking.py"

- Total chunks: 53
- First chunk: LangChain is a framework for developing applications powered by large language mo

PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama>



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

```

example_embedding.py
example_embedding.py > ...
1 from langchain_ollama import OllamaEmbeddings
2
3 embeddings = OllamaEmbeddings(
4     model="nomic-embed-text",
5 )
6
7 text_to_embed = "LangChain is a framework for developing applications powered by large language mo
8
9 single_vector = embeddings.embed_query(text_to_embed)
10 print(str(single_vector))

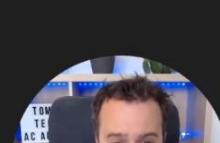
```



The screenshot shows a code editor with a dark theme. On the left is a file tree containing files like `example_embedding.py`, `example_retriever.py`, `keywords.xlsx`, `readme.md`, `requirements.txt`, and `thumbnail_small.png`. The main pane displays the following Python code:

```
    6
    7     = "LangChain is a framework for developing applications powered by large language models (LLMs)."
    8
    9     = embeddings.embed_query(text_to_embed)
10     gle_vector))
```

```
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/example_embedding.py"
```



A screenshot of a code editor showing a file named ".env" with environment variable definitions. The variables include EMBEDDING_MODEL, CHAT_MODEL, MODEL_PROVIDER, CHAT_MODEL, MODEL_PROVIDER, OPENAI_API_KEY, ANTHROPIC_API_KEY, BRIGHTDATA_API_KEY, and various storage and database settings.

```
# == OLLAMA ==
EMBEDDING_MODEL = "mxbai-embed-large"
CHAT_MODEL = "llama3.2:3b"
MODEL_PROVIDER = "ollama"

# == OPENAI ==
#CHAT_MODEL = "gpt-4o-mini"
#MODEL_PROVIDER = "openai"
#OPENAI_API_KEY = "sk-"

# == ANTHROPIC ==
#CHAT_MODEL = "claude-3-7-sonnet-20250219"
#MODEL_PROVIDER = "anthropic"
#ANTHROPIC_API_KEY = "anthropic-"

# == BRIGHTDATA ==
BRIGHTDATA_API_KEY = "[YOUR API KEY HERE]"

# == ENV VARS ==
DATASET_STORAGE_FOLDER = "datasets/"
SNAPSHOT_STORAGE_FILE = "snapshot.txt"

# == CHROMA COLLECTION NAME ==
DATABASE_LOCATION = "chroma_db"
COLLECTION_NAME = "rag_data"
```

Next, let us build a RAG pipeline with our local data

Screenshot of Microsoft Excel showing a table with two rows of data:

	Keyword	Pages
1	python	1
2	langchain	1
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

```

File Edit Selection View Go Run Terminal Help
EXPLORER LOCAL-RAG-WITH-OLLAMA .env .env
> venv
> .env
1 # == OLLAMA ==
2 EMBEDDING_MODEL = "mxbai-embed-large"
3 CHAT_MODEL = "llama3.2:3b"
4 MODEL_PROVIDER = "ollama"
5
6 # == OPENAI ==
7 #CHAT_MODEL = "gpt-4o-mini"
8 #MODEL_PROVIDER = "openai"
9 #OPENAI_API_KEY = "sk-"
10
11 # == ANTHROPIC ==
12 #CHAT_MODEL = "claude-3-7-sonnet-20250219"
13 #MODEL_PROVIDER = "anthropic"
14 #ANTHROPIC_API_KEY = "anthropic-"
15
16 # == BRIGHTDATA ==
17 BRIGHTDATA_API_KEY = "[YOUR API KEY HERE]"
18
19 # == ENV VARS ==
20 DATASET_STORAGE_FOLDER = "datasets/"
21 SNAPSHOT_STORAGE_FILE = "snapshot.txt"
22
23 # == CHROMA COLLECTION NAME ==
24 DATABASE_LOCATION = "chroma_db"
25 COLLECTION_NAME = "rag_data"

```

Bright AI - Ask anything - e.g. "How do I configure proxies in specific country?"

Proxies & Scraping Infrastructure

My Zones Dashboard Event Log More

Zone	Type	Cost	Usage/Spend Limit	Traffic (MB)	Requests	Spent	Status
residential_proxy4	Residential	\$8.4/GB	\$200/day				Active
scraping_browser2	Browser API	Regular: \$8.4/GB Premium: \$11.4/GB	\$200/day				Active
scraping_browser3	Browser API	Regular: \$8.4/GB Premium: \$11.4/GB	No limit				Active
serp_api1	SERP API	\$1.5/CPM	No limit	24 reqs	\$0	Active	
Count: 4			Traffic B	Requests 24	Sum \$0		

Bright AI - Ask anything - e.g. "How do I configure proxies in specific country?"

Web Scrapers

Public web data

My scrapers Logs Web Scrapers Library Tickets

Documentation	FAQ	Scrapers name	Type	Records	Status	Last delivered
		Wikipedia articles - discover by keyword	</> Scraper API	107	Ready	22-May-2025
		Crawl API - discover by domain url	</> Scraper API	132	Ready	09-May-2025
		Crawl API - collect by URL	</> Scraper API	1	Ready	28-Apr-2025
		Indeed job listings information - discover by keyword	</> Scraper API	7	Ready	24-Mar-2025
		LinkedIn people profiles - collect by URL	</> Scraper API	1	Ready	12-Dec-2024
		Zillow properties listing information - collect by URL	</> Scraper API	1	Ready	16-Oct-2024
Count: 6						

Bright Data - Web Data Platform

brightdata.com/cp/scrapers/browse?category=ad_verification&id=hl_208ad069

Web Scrapers Library

Choose your target domain. Set your parameters, and start collecting data!

Search domains

Discover scrapers for breaking news, media trends, and sentiment analysis, perfect for market insights, public opinion tracking, and competitive intelligence.

All Categories Most popular AI Search Business (B2B) Data for AI E-commerce Financial Marketplace News Real-estate Social media Travel

Marketplace 4 domains

- gp play.google.com** Extract data about Android apps, developer details, user reviews, and more from play.google.com. play.google.com 2 scrapers 1.1K
- as apps.apple.com** Extract data about iOS applications, developer information, user reviews, ratings, and more from apps.apple.com. apps.apple.com 2 scrapers 627
- ol olx.com** Extract data about classified ads, product listings, and seller details from olx.com. olx.com 1 scraper 420
- ya yapo.cl** Extract data about classified ads, vehicle listings, property listings, and more from yapo.cl. yapo.cl 1 scraper 313



Bright Data - Web Data Platform

brightdata.com/cp/scrapers/browse?category=all&id=hl_208ad069

Web Scrapers Library

Choose your target domain. Set your parameters, and start collecting data!

wikij

Domains 1 en.wikipedia.org

All Categories Most popular Request custom scraper or Build your own web scraper to get exactly the data you need Documentation FAQ

Social media Travel

Most popular

- in linkedin.com** Extract data about profiles, company pages, job postings, connections, and more from linkedin.com. linkedin.com 11 scrapers 53K
- in instagram.com** Extract data about user profiles, posts, images, videos, followers, following, and more from instagram.com. instagram.com 7 scrapers 12.4K
- tt tiktok.com** Extract data about videos, users, and channels based on hashtags or scrape full user profiles including posts, total likes, ... tiktok.com 12 scrapers 6.8K
- fa facebook.com** Extract data about profiles, posts, groups, events, marketplace, and more from facebook.com. facebook.com 12 scrapers 8.7K
- x x.com** Extract data about posts, profiles, likes, and more from x.com. x.com 3 scrapers 4.8K
- cg chatgpt.com** Extract search results and citations from Chat GPT. chatgpt.com



Bright Data - Web Data Platform

brightdata.com/cp/scrapers/browse?domain=en.wikipedia.org&id=hl_208ad069

My scrapers / Web Scrapers Library / Data for AI / en.wikipedia.org

en.wikipedia.org

Extract data about articles, categories, and contributors from en.wikipedia.org.

Public web data

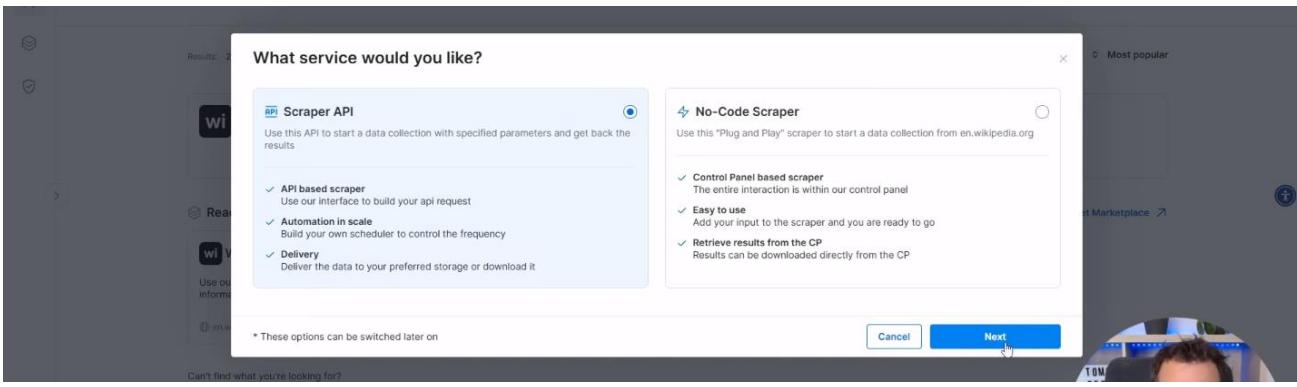
Results: 2 scrapers

Most popular

- Wikipedia articles - collect by URL** Collect Wikipedia articles by URL en.wikipedia.org 450
- Wikipedia articles - discover by keyword** Discover new articles by searching keywords in Wikipedia en.wikipedia.org 450

Ready-to-use Datasets 1 datasets

Browse Dataset Marketplace



The screenshot shows the "Available Scrapers" section of the Bright Data platform. Under "Wikipedia articles", the "Discover by keyword" option is selected. A message indicates that no API key is available.

Trigger Data Collection API

Use this API to start a data collection with specified parameters. It will return the "snapshot_id" for reference.

Add inputs (JSON) keyword: required pages_load: required

amazon 1
apple 5

Count: 2

API Key: API_TOKEN

curl -H "Authorization: Bearer API_TOKEN" -H "Content-Type: application/json" -d '[{"keyword": "amazon", "pages_load": 1}, {"keyword": "apple", "pages_load": 5}]' "https://api.brightdata.com/datasets/v3/trigger?dataset_id=gd_lr978962kkj3nx49&include_errors=true&type=discover"

Description

Discover new articles by searching keywords in Wikipedia

en.wikipedia.org
public web data
450

Source Required

Required inputs to scrape Wikipedia articles - collect by URL:

keyword string Required
Search keyword in Wikipedia

pages_load number Required
Number of pages to load; each page contains up to 20 articles

The screenshot shows the "Users" and "API keys" sections. In the "API keys" section, a new API key is being created for the user "thomas.janssen@tomstechacademy.com".

Add API key

User: thomas.janssen@tomstechacademy.com

Permissions: User

Expiration date: 21/08/2025

Unlimited

2-step verification: Off

Password reset reminder: Off

Download CSV

New user

Cancel Save

The screenshot shows the "API keys" section. A new API key has been generated and copied to the clipboard.

New API key

This is your new API key, please copy it and keep it somewhere safe.

890

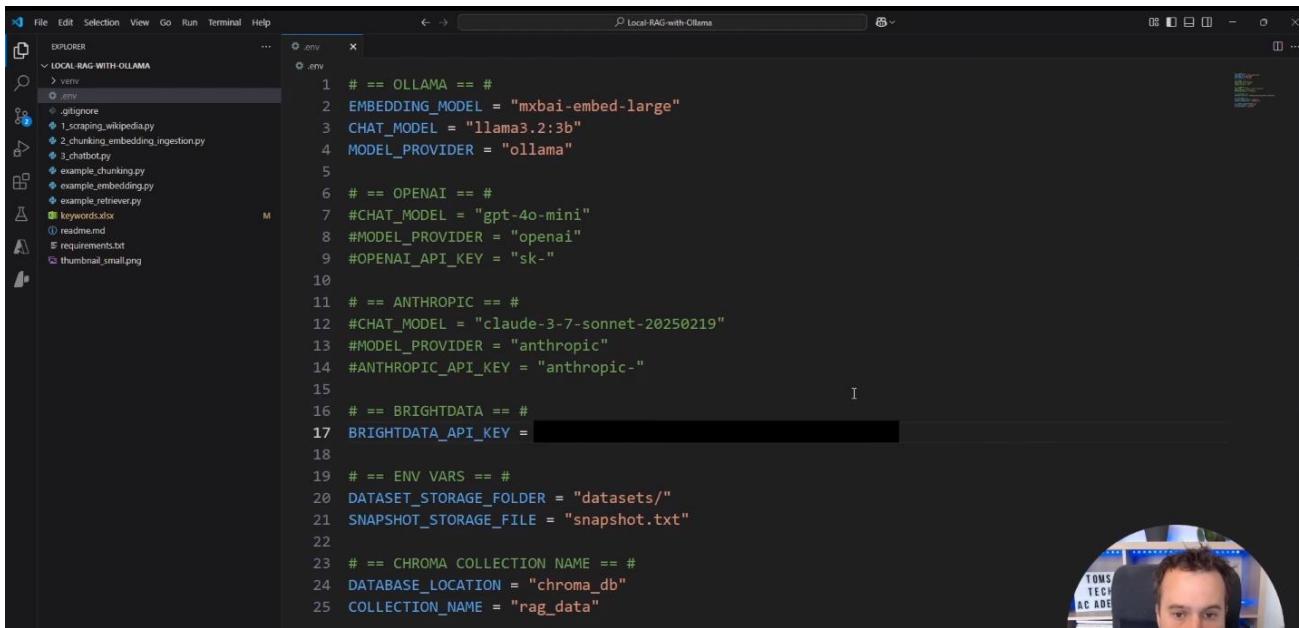
Copied to clipboard

Got it

Expiration date: 21/08/2025

2-step verification: Off

Password reset reminder: Off



```
# == OLLAMA ==
EMBEDDING_MODEL = "mxbai-embed-large"
CHAT_MODEL = "llama3.2:3b"
MODEL_PROVIDER = "ollama"

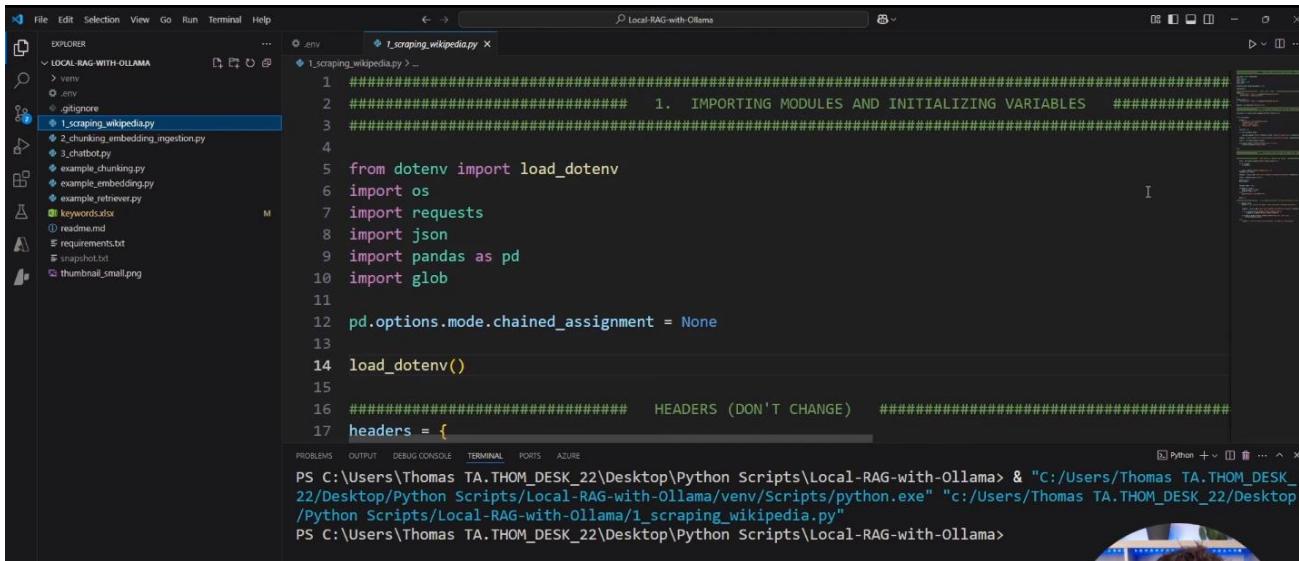
# == OPENAI ==
#CHAT_MODEL = "gpt-4o-mini"
#MODEL_PROVIDER = "openai"
#OPENAI_API_KEY = "sk-"

# == ANTHROPIC ==
#CHAT_MODEL = "claude-3-7-sonnet-20250219"
#MODEL_PROVIDER = "anthropic"
#ANTHROPIC_API_KEY = "anthropic-"

# == BRIGHTDATA ==
BRIGHTDATA_API_KEY = [REDACTED]

# == ENV VARS ==
DATASET_STORAGE_FOLDER = "datasets/"
SNAPSHOT_STORAGE_FILE = "snapshot.txt"

# == CHROMA COLLECTION NAME ==
DATABASE_LOCATION = "chroma_db"
COLLECTION_NAME = "rag_data"
```



```
#####
##### 1. IMPORTING MODULES AND INITIALIZING VARIABLES #####
#####

from dotenv import load_dotenv
import os
import requests
import json
import pandas as pd
import glob

pd.options.mode.chained_assignment = None

load_dotenv()

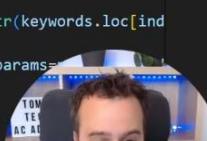
#####
##### HEADERS (DON'T CHANGE) #####
#####

headers = {
```

PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/1_scraping_wikipedia.py"

PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama>

Execute this script that sends all our interested keywords to the BrightData API endpoint for scraping Wikipedia pages



The screenshot shows a code editor interface with a Python script named `1_scraping_wikipedia.py` open. The script is designed to trigger a dataset snapshot creation based on keywords. It uses the Bright Data API to post a trigger and then writes the resulting snapshot ID to a file.

```
File Edit Selection View Go Run Terminal Help
LOCAL-RAG-WITH-Ollama
> venv
@ .env
.gitignore
1_scraping_wikipedia.py
2_chunking_embedding_ingestion.py
3_chartbot.py
example_chunking.py
example_embedding.py
example_refresher.py
keywords.xlsx
readme.md
requirements.txt
snapshot.txt
thumbnail_small.png

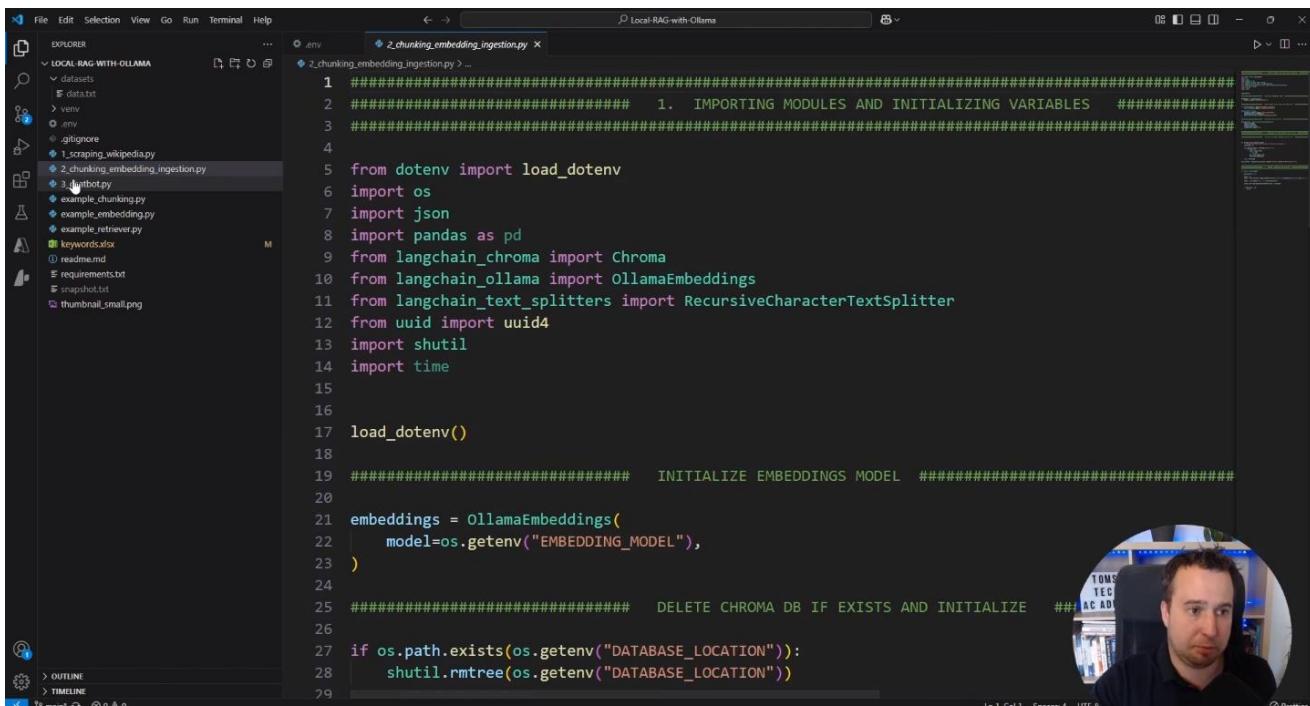
1_scraping_wikipedia.py ...
2. IF SnapshotID IS NOT SET IN .XLSX FILE, TRIGGER CREATION OF
#####
29 #####
30 #####
31 #####
32 file_exists = os.path.isfile(os.getenv("SNAPSHOT_STORAGE_FILE"))
33 #####
34 #####
35 if not file_exists:
36 #####
37     params = {
38         "dataset_id": "gd_lr9978962kkjr3nx49",
39         "include_errors": "true",
40         "type": "discover_new",
41         "discover_by": "keyword",
42     }
43 #####
44 json_data = []
45 #####
46 for ind in keywords.index:
47     json_data.append({"keyword":keywords.loc[ind, "Keyword"], "pages_load":str(keywords.loc[ind, "Pages_Load"])}
48 #####
49 response = requests.post('https://api.brightdata.com/datasets/v3/trigger', params=params)
50 #####
51 result = json.loads(response.content)
52 #####
53 #####
54 with open(os.getenv("SNAPSHOT_STORAGE_FILE"), "a") as f:
55     f.write(str(result["snapshot_id"]))
56 #####
57 #####
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama & "C:/Users/Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama\venv\Scripts\python.exe" "c:/Users/Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama\1_scraping_wikipedia.py"
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama & "C:/Users/Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama\venv\Scripts\python.exe" "c:/Users/Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama\1_scraping_wikipedia.py"
status
ready
Snapshot is ready

==> All articles are ready - start writing data to datasets directory
```

This file contains all the scraped data. we can now chunk, embedding and ingest this data into our ChromaDB vectorDB



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

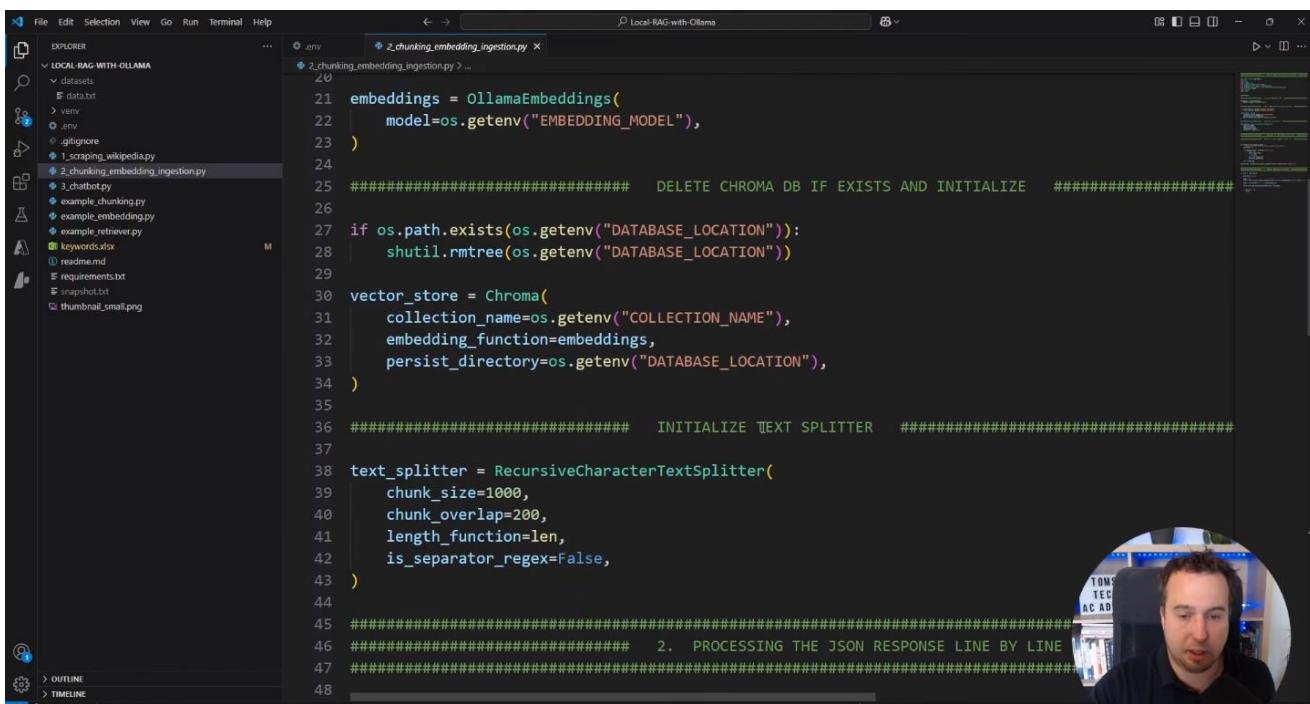
- datasets
- env
- .gitignore
- 1_scraping_wikipedia.py
- 2_chunking_embedding_ingestion.py
- 3_chatbot.py
- example_chunking.py
- example_embedding.py
- example_retriever.py
- keywords.xlsx
- readme.md
- requirements.txt
- snapshot.tst
- thumbnail_small.png

OUTLINE TIMELINE

Local-RAG-with-Ollama

2_chunking_embedding_ingestion.py

```
1 ##### 1. IMPORTING MODULES AND INITIALIZING VARIABLES #####
2 ##### 1. IMPORTING MODULES AND INITIALIZING VARIABLES #####
3 #####
4
5 from dotenv import load_dotenv
6 import os
7 import json
8 import pandas as pd
9 from langchain_chroma import Chroma
10 from langchain_ollama import OllamaEmbeddings
11 from langchain_text_splitters import RecursiveCharacterTextSplitter
12 from uuid import uuid4
13 import shutil
14 import time
15
16
17 load_dotenv()
18
19 ##### INITIALIZE EMBEDDINGS MODEL #####
20
21 embeddings = OllamaEmbeddings(
22     model=os.getenv("EMBEDDING_MODEL"),
23 )
24
25 ##### DELETE CHROMA DB IF EXISTS AND INITIALIZE #####
26
27 if os.path.exists(os.getenv("DATABASE_LOCATION")):
28     shutil.rmtree(os.getenv("DATABASE_LOCATION"))
29
```



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

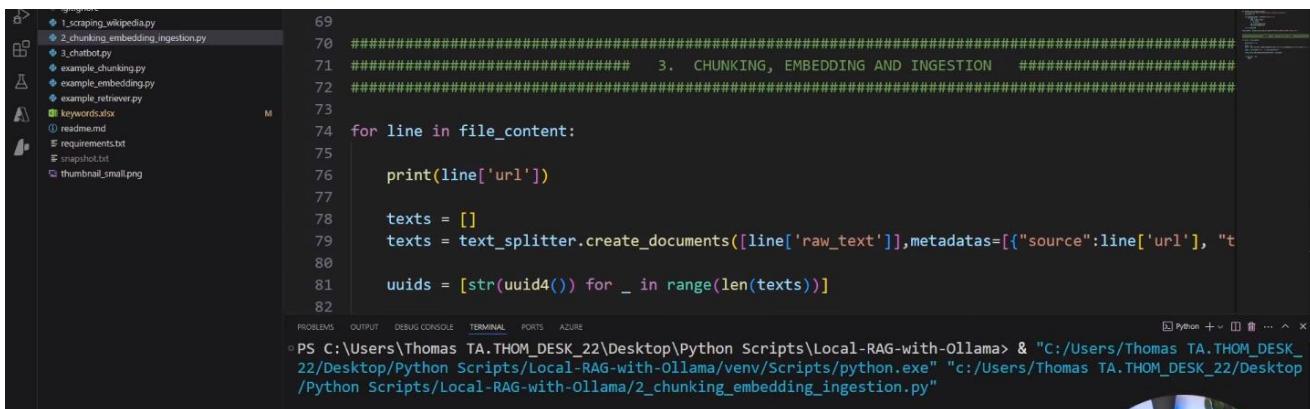
- datasets
- env
- .gitignore
- 1_scraping_wikipedia.py
- 2_chunking_embedding_ingestion.py
- 3_chatbot.py
- example_chunking.py
- example_embedding.py
- example_retriever.py
- keywords.xlsx
- readme.md
- requirements.txt
- snapshot.tst
- thumbnail_small.png

OUTLINE TIMELINE

Local-RAG-with-Ollama

2_chunking_embedding_ingestion.py

```
20
21 embeddings = OllamaEmbeddings(
22     model=os.getenv("EMBEDDING_MODEL"),
23 )
24
25 ##### DELETE CHROMA DB IF EXISTS AND INITIALIZE #####
26
27 if os.path.exists(os.getenv("DATABASE_LOCATION")):
28     shutil.rmtree(os.getenv("DATABASE_LOCATION"))
29
30 vector_store = Chroma(
31     collection_name=os.getenv("COLLECTION_NAME"),
32     embedding_function=embeddings,
33     persist_directory=os.getenv("DATABASE_LOCATION"),
34 )
35
36 ##### INITIALIZE TEXT SPLITTER #####
37
38 text_splitter = RecursiveCharacterTextSplitter(
39     chunk_size=1000,
40     chunk_overlap=200,
41     length_function=len,
42     is_separator_regex=False,
43 )
44
45 ##### 2. PROCESSING THE JSON RESPONSE LINE BY LINE #####
46
47
48
```



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

- datasets
- env
- .gitignore
- 1_scraping_wikipedia.py
- 2_chunking_embedding_ingestion.py
- 3_chatbot.py
- example_chunking.py
- example_embedding.py
- example_retriever.py
- keywords.xlsx
- readme.md
- requirements.txt
- snapshot.tst
- thumbnail_small.png

OUTLINE TIMELINE

Local-RAG-with-Ollama

2_chunking_embedding_ingestion.py

```
69
70 ##### 3. CHUNKING, EMBEDDING AND INGESTION #####
71 #####
72 #####
73
74 for line in file_content:
75
76     print(line['url'])
77
78     texts = []
79     texts = text_splitter.create_documents([line['raw_text']], metadatas=[{"source":line['url'], "t
80
81     uuids = [str(uuid4()) for _ in range(len(texts))]
82
```

PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/2_chunking_embedding_ingestion.py"



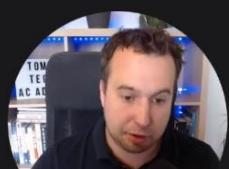
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
https://en.wikipedia.org/wiki/Intelligent_agent
https://en.wikipedia.org/wiki/Sentence_embedding
https://en.wikipedia.org/wiki/Milvus_(vector_database)
https://en.wikipedia.org/wiki/DataStax
https://en.wikipedia.org/wiki/Reticulated_python
https://en.wikipedia.org/wiki/Python_(codename)
https://en.wikipedia.org/wiki/Burmese_python
https://en.wikipedia.org/wiki/Python
https://en.wikipedia.org/wiki/Monty_Python
https://en.wikipedia.org/wiki/Python_(programming_language)
https://en.wikipedia.org/wiki/Monty_Python_and_the_Holy_Grail
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
https://en.wikipedia.org/wiki/Milvus_(vector_database)
https://en.wikipedia.org/wiki/DataStax
https://en.wikipedia.org/wiki/Reticulated_python
https://en.wikipedia.org/wiki/Python_(codename)
https://en.wikipedia.org/wiki/Burmese_python
https://en.wikipedia.org/wiki/Python
https://en.wikipedia.org/wiki/Monty_Python
https://en.wikipedia.org/wiki/Python_(programming_language)
https://en.wikipedia.org/wiki/Monty_Python_and_the_Holy_Grail
https://en.wikipedia.org/wiki/History_of_Python
https://en.wikipedia.org/wiki/Indian_python
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
https://en.wikipedia.org/wiki/Python_(genus)
https://en.wikipedia.org/wiki/Python_Conference
https://en.wikipedia.org/wiki/Central_African_rock_python
https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)
https://en.wikipedia.org/wiki/Green_tree_python
https://en.wikipedia.org/wiki/Zen_of_Python
https://en.wikipedia.org/wiki/Mod_python
https://en.wikipedia.org/wiki/Burmese_pythons_in_Florida
https://en.wikipedia.org/wiki/Monty_Python%27s_The_Meaning_of_Life
https://en.wikipedia.org/wiki/Robin_Williams
https://en.wikipedia.org/wiki/Python_compiler
```

PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama>

Thomas Janssen (1 day ago) · Ln 87, Col 14 · Spaces: 4 · UTF-8 · prettier

All the pages have been added to the DB



File Edit Selection View Go Run Terminal Help

LOCAL-RAG-WITH-OLLAMA

EXPLORER .env example_retriever.py

chroma.db datasets venv .env .gitignore 1_scraping_wikipedia.py 2_chunking_embedding_ingestion.py 3_chatbot.py example_chunking.py example_embedding.py example_retriever.py keywords.xlsx README.md requirements.txt snapshot.txt thumbnail_small.png

example_retriever.py > -

```
1 ##### 1. IMPORTING MODULES AND INITIALIZING VARIABLES #####
2 ##########
3 #####
4
5 from dotenv import load_dotenv
6 import os
7 import pandas as pd
8 from langchain_chroma import Chroma
9 from langchain_ollama import OllamaEmbeddings
10
11
12 pd.options.mode.chained_assignment = None
13
14 load_dotenv()
15
16 ########## INITIALIZE EMBEDDINGS MODEL #####
17
18 embeddings = OllamaEmbeddings(
19     model=os.getenv("EMBEDDING_MODEL"),
20 )
21
22 ########## INITIALIZE CHROMA VECTOR STORE #####
23
24 vector_store = Chroma(
25     collection_name=os.getenv("COLLECTION_NAME"),
26     embedding_function=embeddings,
27     persist_directory=os.getenv("DATABASE_LOCATION"),
28 )
```

A screenshot of the Visual Studio Code interface. The left sidebar shows a project structure for 'LOCAL-RAG-WITH-OLLAMA' containing files like .env, README.md, requirements.txt, and various Python scripts. The main editor area displays a Python script named 'example_retriever.py'. The code initializes an embeddings model using Ollama and a Chroma vector store, then performs a similarity search on the query 'what is langchain'. A circular video overlay in the bottom right corner shows a person's face.

```
14 load_dotenv()
15
16 ##### INITIALIZED EMBEDDINGS MODEL #####
17
18 embeddings = OllamaEmbeddings(
19     model=os.getenv("EMBEDDING_MODEL"),
20 )
21
22 ##### INITIALIZED CHROMA VECTOR STORE #####
23
24 vector_store = Chroma(
25     collection_name=os.getenv("COLLECTION_NAME"),
26     embedding_function=embeddings,
27     persist_directory=os.getenv("DATABASE_LOCATION"),
28 )
29
30 results = vector_store.similarity_search_by_vector(
31     embedding=embeddings.embed_query("what is langchain"), k=5
32 )
33
34 for doc in results:
35     print(f"* {doc.page_content} [{doc.metadata}]")
```

A screenshot of the VS Code interface focusing on the terminal tab. The command PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/example_retriever.py" is being run. The terminal output shows the results of the code execution.

A screenshot of the VS Code interface focusing on the terminal tab. The command PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/example_retriever.py" is being run. The terminal output shows the results of the code execution.

A screenshot of the VS Code interface focusing on the terminal tab. The command PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama & "C:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/venv/Scripts/python.exe" "c:/Users/Thomas TA.THOM_DESK_22/Desktop/Python Scripts/Local-RAG-with-Ollama/example_retriever.py" is being run. The terminal output shows the results of the code execution.



A screenshot of a code editor window titled "Local RAG-with-Ollama". The left sidebar shows a project structure under "LOCAL-RAG-WITH-OLLAMA" with files like .env, 1_scraping_wikipedia.py, 2_chunking_embedding_ingestion.py, 3_chatbot.py, example_chunking.py, example_embedding.py, example_retriever.py, keywords.txt, readme.md, requirements.txt, snapshot.txt, and thumbnail_small.png. The main editor area displays Python code for initializing an embeddings model and a chroma vector store using LangChain and Ollama. The code includes imports for basics, os, dotenv, streamlit, langchain, langchain_chroma, langchain_ollama, langchain.chat_models, langchain_core.messages, langchain.agents, langchain import hub, langchain_core.prompts, langchain_core.tools, and langchain.agents.AgentExecutor. It loads environment variables from .env and initializes the embeddings model with OllamaEmbeddings and the specified embedding model. It also initializes the chroma vector store.

```
1 # import basics
2 import os
3 from dotenv import load_dotenv
4
5 # import streamlit
6 import streamlit as st
7
8 # import langchain
9 from langchain.agents import AgentExecutor
10 from langchain_chroma import Chroma
11 from langchain_ollama import OllamaEmbeddings
12 from langchain.chat_models import init_chat_model
13 from langchain_core.messages import AIMessage, HumanMessage
14 from langchain.agents import create_tool_calling_agent
15 from langchain import hub
16 from langchain_core.prompts import PromptTemplate
17 from langchain_core.tools import tool
18
19 # load environment variables
20 load_dotenv()
21
22 ##### INITIATE EMBEDDINGS MODEL #####
23
24 embeddings = OllamaEmbeddings(
25     model=os.getenv("EMBEDDING_MODEL"),
26 )
27
28 ##### INITIATE CHROMA VECTOR STORE #####
29
```



A screenshot of a code editor window titled "Local RAG-with-Ollama". The left sidebar shows a project structure under "LOCAL-RAG-WITH-OLLAMA" with files like .env, 1_scraping_wikipedia.py, 2_chunking_embedding_ingestion.py, 3_chatbot.py, example_chunking.py, example_embedding.py, example_retriever.py, keywords.txt, readme.md, requirements.txt, snapshot.txt, and thumbnail_small.png. The main editor area displays a series of numbered instructions for interacting with the chatbot. It starts by introducing itself as a helpful assistant and providing a query and chat history. It explains how to use the 'retrieve' tool to get relevant information. It then asks for the query, chat history, and source information. It also mentions the scratchpad for intermediate results and provides a template for returning text, including an answer, source URL, and a blank line. Finally, it describes the creation of a retriever tool.

```
50 You are a helpful assistant. You will be provided with a query and a chat history.
51 Your task is to retrieve relevant information from the vector store and provide a response.
52 For this you use the tool 'retrieve' to get the relevant information.
53
54 The query is as follows:
55 {input}
56
57 The chat history is as follows:
58 {chat_history}
59
60 Please provide a concise and informative response based on the retrieved information.
61 If you don't know the answer, say "I don't know" (and don't provide a source).
62
63 You can use the scratchpad to store any intermediate results or notes.
64 The scratchpad is as follows:
65 {agent_scratchpad}
66
67 For every piece of information you provide, also provide the source.
68
69 Return text as follows:
70 I
71 <Answer to the question>
72 Source: source_url
73 """
74
75
76 # creating the retriever tool
77 @tool
```



```
example_retriever.py
keywords.xlsx
readme.md
requirements.txt
snapshot.b6t
thumbnail_small.png

76 # creating the retriever tool
77 @tool
78 def retrieve(query: str):
79     """Retrieve information related to a query."""
80     retrieved_docs = vector_store.similarity_search(query, k=2)
81
82     serialized = ""
83
84     for doc in retrieved_docs:
85         serialized += f"Source: {doc.metadata['source']}\nContent: {doc.page_content}\n\n"
86
87     return serialized
88
89 # combining all tools
90 tools = [retrieve]
91
92 # initiating the agent
93 agent = create_tool_calling_agent(llm, tools, prompt)
94
95 # create the agent executor
96 agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True)
97
```

Thomas Janssen (1 day ago) · In 88 Col 1 · Spaces: 4 · UTF-8



```
thumbnail_small.png

95 # create the agent executor
96 agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True)
97
98 # initiating streamlit app
99 st.set_page_config(page_title="Agentic RAG Chatbot", page_icon="🤖")
100 st.title("🤖 Agentic RAG Chatbot")
101
102 # initialize chat history
103 if "messages" not in st.session_state:
104     st.session_state.messages = []
105
106 # display chat messages from history on app rerun
```



```
example_chunting.py
example_embedding.py
keywords.xlsx
readme.md
requirements.txt
snapshot.b6t
thumbnail_small.png

106 # display chat messages from history on app rerun
107 for message in st.session_state.messages:
108     if isinstance(message, HumanMessage):
109         with st.chat_message("user"):
110             st.markdown(message.content)
111     elif isinstance(message, AIMessage):
112         with st.chat_message("assistant"):
113             st.markdown(message.content)
114
115
116 # create the bar where we can type messages
117 user_question = st.chat_input("How are you?")
118
119
120 # did the user submit a prompt?
121 if user_question:
122
123     # add the message from the user (prompt) to the screen with streamlit
124     with st.chat_message("user"):
125         st.markdown(user_question)
126
127         st.session_state.messages.append(HumanMessage(user_question))
128
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
PS C:\Users\Thomas TA.THOM_DESK_22\Desktop\Python Scripts\Local-RAG-with-Ollama> streamlit run 3_chatbot.py
```

Bright Data - Web Data Platform Agentic RAG Chatbot

localhost:8501 Deploy

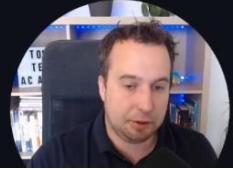
Agentic RAG Chatbot

what is langchain?

LangChain is a software framework that helps facilitate the integration of large language models (LLMs) into applications. As a language model integration framework, LangChain's use-cases largely overlap with those of language models in general, including document analysis and summarization, chatbots, and code analysis.

Source: <https://en.wikipedia.org/wiki/LangChain>

How are you? ➤



Bright Data - Web Data Platform Agentic RAG Chatbot

localhost:8501 Deploy

what is langchain?

LangChain is a software framework that helps facilitate the integration of large language models (LLMs) into applications. As a language model integration framework, LangChain's use-cases largely overlap with those of language models in general, including document analysis and summarization, chatbots, and code analysis.

Source: <https://en.wikipedia.org/wiki/LangChain>

what is python and whom created it?

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically type-checked and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language, and he first released it in 1991 as Python 0.9. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2. Python consistently ranks as one of the most popular programming languages, and it has gained widespread use in the machine learning community.

Source: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

How are you? ➤



This works as expected