Goal is to build a simple CHATBOT I can use to query against a number of GitHub Repos my group has developed for open source projects. I have 10+ repositories related to my software, scattered about. For purposes of this project, we can start with three to index / upload: https://github.com/digitalslidearchive/digital\_slide\_archive https://digitalslidearchive.github.io/digital\_slide\_archive/ https://github.com/DigitalSlideArchive/digitalslidearchive.info Requirements: Provide source code with documentation Key aspects: 1) Updating/ redeploying streamlit app 2) Loading new documents to the system (I’ll be adding more repos over time) 3) Reindexing the system after document update 4) If OpenAPI model changes / updates, what steps are needed to reindex/reencode things? 5) Question: How hard is it to change the backend? Could I use OpenLLAMA and self host the API? If I want to try other sentence/document encoders, is that possible and if so what are the steps?

Building a chatbot to query against multiple GitHub repositories is a great project! Below is a high-level plan to achieve your goal, along with answers to your specific requirements and questions.

**High-Level Plan**

1. **Data Collection**:
   * Clone the GitHub repositories.
   * Extract relevant text/data (e.g., READMEs, code comments, documentation, etc.).
2. **Data Indexing**:
   * Use a document embedding model (e.g., OpenAI embeddings, Sentence Transformers) to encode the text.
   * Store the embeddings in a vector database (e.g., FAISS, Pinecone, Weaviate).
3. **Chatbot Backend**:
   * Use a language model (e.g., OpenAI GPT, OpenLLaMA) to generate responses.
   * Integrate the vector database for semantic search.
4. **Frontend**:
   * Use Streamlit to create a simple web interface for querying the chatbot.
5. **Deployment**:
   * Deploy the Streamlit app using a service like Streamlit Sharing, Heroku, or AWS.

**Key Aspects**

**1) Updating/Redeploying Streamlit App**

* Streamlit apps can be updated by pushing changes to the codebase (e.g., GitHub repository).
* Redeployment is automatic if you use Streamlit Sharing or CI/CD pipelines (e.g., GitHub Actions).

**2) Loading New Documents to the System**

* Add a script to clone new repositories and extract text.
* Update the vector database with new embeddings.
* Example workflow:

python

Copy

def add\_new\_repo(repo\_url):

# Clone repo

repo\_name = repo\_url.split("/")[-1]

os.system(f"git clone {repo\_url}")

# Extract text

text\_data = extract\_text\_from\_repo(repo\_name)

# Generate embeddings

embeddings = embed\_text(text\_data)

# Update vector database

vector\_db.add(embeddings, metadata=repo\_name)

**3) Reindexing the System After Document Update**

* Reindexing involves regenerating embeddings for all documents and updating the vector database.
* Example:

python

Copy

def reindex():

all\_texts = load\_all\_texts()

new\_embeddings = embed\_text(all\_texts)

vector\_db.reindex(new\_embeddings)

**4) If OpenAI Model Changes/Updates**

* If the embedding model changes, you’ll need to re-encode all documents with the new model.
* Steps:
  1. Update the embedding model in your code.
  2. Regenerate embeddings for all documents.
  3. Reindex the vector database.

**5) Changing the Backend**

* **Using OpenLLaMA**:
  + You can replace OpenAI with OpenLLaMA by hosting the model locally or on a cloud service.
  + Steps:
    1. Download and set up OpenLLaMA.
    2. Replace the OpenAI API calls with calls to your OpenLLaMA API.
    3. Ensure the input/output formats match.
* **Using Other Sentence/Document Encoders**:
  + You can replace the embedding model with any compatible encoder (e.g., Sentence Transformers, BERT).
  + Steps:
    1. Install the new encoder library.
    2. Replace the embedding generation code.
    3. Regenerate embeddings and reindex the vector database.

**Implementation Steps**

**1. Set Up the Environment**

* Install required libraries:

bash

Copy

pip install streamlit openai sentence-transformers faiss-cpu gitpython

**2. Clone and Extract Text from Repositories**

* Use gitpython to clone repos and extract text:

python

Copy

from git import Repo

import os

def clone\_repo(repo\_url, output\_dir):

Repo.clone\_from(repo\_url, output\_dir)

def extract\_text\_from\_repo(repo\_dir):

text\_data = ""

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r") as f:

text\_data += f.read()

return text\_data

**3. Generate Embeddings**

* Use a pre-trained model (e.g., OpenAI or Sentence Transformers):

python

Copy

from sentence\_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2")

def embed\_text(text):

return model.encode(text)

**4. Set Up Vector Database**

* Use FAISS for local storage:

python

Copy

import faiss

import numpy as np

dimension = 384 # Dimension of embeddings

index = faiss.IndexFlatL2(dimension)

def add\_to\_index(embeddings):

index.add(np.array(embeddings))

**5. Build the Chatbot**

* Use Streamlit for the frontend:

python

Copy

import streamlit as st

st.title("GitHub Repo Chatbot")

query = st.text\_input("Ask a question:")

if query:

query\_embedding = embed\_text(query)

distances, indices = index.search(np.array([query\_embedding]), k=5)

st.write("Top results:")

for idx in indices[0]:

st.write(text\_data[idx])

**6. Deploy the App**

* Deploy the Streamlit app using Streamlit Sharing or another service.

**Example Code**

Here’s a simplified example:

python

Copy

import os

import numpy as np

import streamlit as st

from git import Repo

from sentence\_transformers import SentenceTransformer

import faiss

# Clone repo

def clone\_repo(repo\_url, output\_dir):

Repo.clone\_from(repo\_url, output\_dir)

# Extract text

def extract\_text\_from\_repo(repo\_dir):

text\_data = ""

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r") as f:

text\_data += f.read()

return text\_data

# Embed text

model = SentenceTransformer("all-MiniLM-L6-v2")

def embed\_text(text):

return model.encode(text)

# Set up FAISS index

dimension = 384

index = faiss.IndexFlatL2(dimension)

# Streamlit app

st.title("GitHub Repo Chatbot")

repo\_url = st.text\_input("Enter GitHub repo URL:")

if repo\_url:

clone\_repo(repo\_url, "temp\_repo")

text\_data = extract\_text\_from\_repo("temp\_repo")

embeddings = embed\_text(text\_data)

index.add(np.array(embeddings))

query = st.text\_input("Ask a question:")

if query:

query\_embedding = embed\_text(query)

distances, indices = index.search(np.array([query\_embedding]), k=5)

st.write("Top results:")

for idx in indices[0]:

st.write(text\_data[idx])

**Next Steps**

1. Expand the system to handle multiple repositories.
2. Add support for more file types (e.g., .py, .html).
3. Experiment with different embedding models and vector databases.
4. Deploy the app and share it with your team.

Let me know if you need further assistance!

2 / 2

explain me this project step by step from scratch with code of each file

Sure! Let’s break this project down step by step, with code for each file. We’ll build a simple chatbot that can query against GitHub repositories. Here's the step-by-step guide:

**Step 1: Set Up the Project Structure**

Create the following folder structure:

Copy

github-chatbot/

│

├── data/ # Stores cloned repositories

├── app.py # Streamlit app

├── clone\_repos.py # Script to clone repositories

├── embed\_text.py # Script to generate embeddings

├── faiss\_index.py # Script to manage FAISS index

└── requirements.txt # Python dependencies

**Step 2: Install Dependencies**

Create a requirements.txt file:

plaintext

Copy

streamlit

gitpython

sentence-transformers

faiss-cpu

numpy

Install the dependencies:

bash

Copy

pip install -r requirements.txt

**Step 3: Clone Repositories**

Create a clone\_repos.py script to clone GitHub repositories into the data/ folder.

python

Copy

# clone\_repos.py

from git import Repo

import os

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

"""

if not os.path.exists(output\_dir):

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

if \_\_name\_\_ == "\_\_main\_\_":

repos = [

"https://github.com/digitalslidearchive/digital\_slide\_archive",

"https://github.com/DigitalSlideArchive/digitalslidearchive.info"

]

for repo\_url in repos:

repo\_name = repo\_url.split("/")[-1]

clone\_repo(repo\_url, os.path.join("data", repo\_name))

Run this script to clone the repositories:

bash

Copy

python clone\_repos.py

**Step 4: Extract Text from Repositories**

Update the clone\_repos.py script to include a function for extracting text from the cloned repositories.

python

Copy

# clone\_repos.py (updated)

import os

def extract\_text\_from\_repo(repo\_dir):

"""

Extract text from markdown and text files in a repository.

"""

text\_data = ""

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r", encoding="utf-8") as f:

text\_data += f.read() + "\n"

return text\_data

if \_\_name\_\_ == "\_\_main\_\_":

repos = [

"https://github.com/digitalslidearchive/digital\_slide\_archive",

"https://github.com/DigitalSlideArchive/digitalslidearchive.info"

]

for repo\_url in repos:

repo\_name = repo\_url.split("/")[-1]

repo\_dir = os.path.join("data", repo\_name)

clone\_repo(repo\_url, repo\_dir)

text\_data = extract\_text\_from\_repo(repo\_dir)

with open(os.path.join(repo\_dir, "extracted\_text.txt"), "w", encoding="utf-8") as f:

f.write(text\_data)

**Step 5: Generate Embeddings**

Create an embed\_text.py script to generate embeddings for the extracted text using a pre-trained model.

python

Copy

# embed\_text.py

from sentence\_transformers import SentenceTransformer

import numpy as np

import os

def embed\_text(text):

"""

Generate embeddings for the input text.

"""

model = SentenceTransformer("all-MiniLM-L6-v2")

return model.encode(text)

if \_\_name\_\_ == "\_\_main\_\_":

# Load extracted text

repo\_name = "digital\_slide\_archive" # Example repo

with open(os.path.join("data", repo\_name, "extracted\_text.txt"), "r", encoding="utf-8") as f:

text\_data = f.read()

# Generate embeddings

embeddings = embed\_text(text\_data)

np.save(os.path.join("data", repo\_name, "embeddings.npy"), embeddings)

Run this script to generate embeddings:

bash

Copy

python embed\_text.py

**Step 6: Set Up FAISS Index**

Create a faiss\_index.py script to manage the FAISS index for semantic search.

python

Copy

# faiss\_index.py

import faiss

import numpy as np

import os

def create\_index(embeddings):

"""

Create a FAISS index for the embeddings.

"""

dimension = embeddings.shape[1]

index = faiss.IndexFlatL2(dimension)

index.add(embeddings)

return index

if \_\_name\_\_ == "\_\_main\_\_":

# Load embeddings

repo\_name = "digital\_slide\_archive" # Example repo

embeddings = np.load(os.path.join("data", repo\_name, "embeddings.npy"))

# Create and save FAISS index

index = create\_index(embeddings)

faiss.write\_index(index, os.path.join("data", repo\_name, "index.faiss"))

Run this script to create the FAISS index:

bash

Copy

python faiss\_index.py

**Step 7: Build the Streamlit App**

Create an app.py script for the chatbot interface.

python

Copy

# app.py

import streamlit as st

import numpy as np

import faiss

from sentence\_transformers import SentenceTransformer

import os

# Load FAISS index and embeddings

repo\_name = "digital\_slide\_archive" # Example repo

index = faiss.read\_index(os.path.join("data", repo\_name, "index.faiss"))

with open(os.path.join("data", repo\_name, "extracted\_text.txt"), "r", encoding="utf-8") as f:

text\_data = f.read().split("\n")

# Load embedding model

model = SentenceTransformer("all-MiniLM-L6-v2")

# Streamlit app

st.title("GitHub Repo Chatbot")

query = st.text\_input("Ask a question:")

if query:

# Generate query embedding

query\_embedding = model.encode(query)

# Search FAISS index

distances, indices = index.search(np.array([query\_embedding]), k=5)

# Display results

st.write("Top results:")

for idx in indices[0]:

st.write(text\_data[idx])

Run the Streamlit app:

bash

Copy

streamlit run app.py

**Step 8: Deploy the App**

1. Push the project to a GitHub repository.
2. Use a service like [Streamlit Sharing](https://share.streamlit.io/) to deploy the app.

**Summary**

* **clone\_repos.py**: Clones repositories and extracts text.
* **embed\_text.py**: Generates embeddings for the text.
* **faiss\_index.py**: Creates and manages the FAISS index.
* **app.py**: Streamlit app for querying the chatbot.

This is a basic implementation. You can expand it to handle multiple repositories, support more file types, and improve the user interface. Let me know if you need further assistance!

(ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python embed\_text.py 2025-02-03 14:59:26.315702: E external/local\_xla/xla/stream\_executor/cuda/cuda\_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered WARNING: All log messages before absl::InitializeLog() is called are written to STDERR E0000 00:00:1738612766.335412 159213 cuda\_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered E0000 00:00:1738612766.341521 159213 cuda\_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered 2025-02-03 14:59:26.362348: I tensorflow/core/platform/cpu\_feature\_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations. To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags. Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/embed\_text.py", line 16, in <module> with open(os.path.join("data", repo\_name, "extracted\_text.txt"), "r", encoding="utf-8") as f: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ FileNotFoundError: [Errno 2] No such file or directory: 'data/digital\_slide\_archive/extracted\_text.txt' (ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python embed\_text.py 2025-02-03 14:59:40.800509: E external/local\_xla/xla/stream\_executor/cuda/cuda\_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered WARNING: All log messages before absl::InitializeLog() is called are written to STDERR E0000 00:00:1738612780.820580 159447 cuda\_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered E0000 00:00:1738612780.826755 159447 cuda\_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered 2025-02-03 14:59:40.848055: I tensorflow/core/platform/cpu\_feature\_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations. To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags. Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/embed\_text.py", line 16, in <module> with open(os.path.join("data", repo\_name, "extracted\_text.txt"), "r", encoding="utf-8") as f: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ FileNotFoundError: [Errno 2] No such file or directory: 'data/digital\_slide\_archive/extracted\_text.txt' (ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python faiss\_index.py Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/faiss\_index.py", line 18, in <module> embeddings = np.load(os.path.join("data", repo\_name, "embeddings.npy")) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/numpy/lib/\_npyio\_impl.py", line 455, in load fid = stack.enter\_context(open(os.fspath(file), "rb")) ^^^^^^^^^^^^^^^^^^^^^^^^^^^ FileNotFoundError: [Errno 2] No such file or directory: 'data/digital\_slide\_archive/embeddings.npy'

The errors you're encountering indicate that the required files (extracted\_text.txt and embeddings.npy) are missing. This is likely because the earlier steps in the pipeline were not executed successfully. Let's go through the process step by step to ensure everything is set up correctly.

**Step-by-Step Debugging and Fixes**

**1. Verify Repository Cloning**

The error FileNotFoundError: [Errno 2] No such file or directory: 'data/digital\_slide\_archive/extracted\_text.txt' suggests that the repository was not cloned or the text extraction step failed.

* **Check if the repository was cloned**:  
  Run the following command to verify if the repository exists in the data/ folder:

bash

Copy

ls data/digital\_slide\_archive

If the folder is empty or doesn't exist, the cloning step failed.

* **Re-run the cloning script**:  
  Ensure the clone\_repos.py script is executed correctly:

bash

Copy

python clone\_repos.py

* **Verify the output**:  
  After running the script, check if the extracted\_text.txt file is created:

bash

Copy

ls data/digital\_slide\_archive

**2. Fix the Cloning Script**

If the repository is not cloned or the text extraction fails, update the clone\_repos.py script to handle errors and ensure the extracted\_text.txt file is created.

python

Copy

# clone\_repos.py (updated with error handling)

from git import Repo

import os

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

"""

try:

if not os.path.exists(output\_dir):

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

except Exception as e:

print(f"Failed to clone {repo\_url}: {e}")

def extract\_text\_from\_repo(repo\_dir):

"""

Extract text from markdown and text files in a repository.

"""

text\_data = ""

try:

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r", encoding="utf-8") as f:

text\_data += f.read() + "\n"

except Exception as e:

print(f"Failed to extract text from {repo\_dir}: {e}")

return text\_data

if \_\_name\_\_ == "\_\_main\_\_":

repos = [

"https://github.com/digitalslidearchive/digital\_slide\_archive",

"https://github.com/DigitalSlideArchive/digitalslidearchive.info"

]

for repo\_url in repos:

repo\_name = repo\_url.split("/")[-1]

repo\_dir = os.path.join("data", repo\_name)

clone\_repo(repo\_url, repo\_dir)

text\_data = extract\_text\_from\_repo(repo\_dir)

if text\_data:

with open(os.path.join(repo\_dir, "extracted\_text.txt"), "w", encoding="utf-8") as f:

f.write(text\_data)

print(f"Extracted text saved to {os.path.join(repo\_dir, 'extracted\_text.txt')}")

else:

print(f"No text extracted from {repo\_dir}")

**3. Re-run the Cloning Script**

Execute the updated script:

bash

Copy

python clone\_repos.py

Verify that the extracted\_text.txt file is created:

bash

Copy

ls data/digital\_slide\_archive

**4. Generate Embeddings**

Once the extracted\_text.txt file is created, run the embed\_text.py script to generate embeddings:

bash

Copy

python embed\_text.py

If the script fails, ensure the extracted\_text.txt file exists and contains text.

**5. Fix the Embedding Script**

If the embed\_text.py script fails, update it to handle errors and verify the input file:

python

Copy

# embed\_text.py (updated with error handling)

from sentence\_transformers import SentenceTransformer

import numpy as np

import os

def embed\_text(text):

"""

Generate embeddings for the input text.

"""

model = SentenceTransformer("all-MiniLM-L6-v2")

return model.encode(text)

if \_\_name\_\_ == "\_\_main\_\_":

repo\_name = "digital\_slide\_archive" # Example repo

text\_file = os.path.join("data", repo\_name, "extracted\_text.txt")

if not os.path.exists(text\_file):

print(f"Error: {text\_file} does not exist. Run clone\_repos.py first.")

else:

try:

with open(text\_file, "r", encoding="utf-8") as f:

text\_data = f.read()

if not text\_data:

print(f"Error: {text\_file} is empty.")

else:

# Generate embeddings

embeddings = embed\_text(text\_data)

np.save(os.path.join("data", repo\_name, "embeddings.npy"), embeddings)

print(f"Embeddings saved to {os.path.join('data', repo\_name, 'embeddings.npy')}")

except Exception as e:

print(f"Failed to generate embeddings: {e}")

**6. Re-run the Embedding Script**

Execute the updated script:

bash

Copy

python embed\_text.py

Verify that the embeddings.npy file is created:

bash

Copy

ls data/digital\_slide\_archive

**7. Set Up FAISS Index**

Once the embeddings.npy file is created, run the faiss\_index.py script:

bash

Copy

python faiss\_index.py

**8. Fix the FAISS Index Script**

If the faiss\_index.py script fails, update it to handle errors:

python

Copy

# faiss\_index.py (updated with error handling)

import faiss

import numpy as np

import os

def create\_index(embeddings):

"""

Create a FAISS index for the embeddings.

"""

dimension = embeddings.shape[1]

index = faiss.IndexFlatL2(dimension)

index.add(embeddings)

return index

if \_\_name\_\_ == "\_\_main\_\_":

repo\_name = "digital\_slide\_archive" # Example repo

embeddings\_file = os.path.join("data", repo\_name, "embeddings.npy")

if not os.path.exists(embeddings\_file):

print(f"Error: {embeddings\_file} does not exist. Run embed\_text.py first.")

else:

try:

embeddings = np.load(embeddings\_file)

index = create\_index(embeddings)

faiss.write\_index(index, os.path.join("data", repo\_name, "index.faiss"))

print(f"FAISS index saved to {os.path.join('data', repo\_name, 'index.faiss')}")

except Exception as e:

print(f"Failed to create FAISS index: {e}")

**9. Re-run the FAISS Index Script**

Execute the updated script:

bash

Copy

python faiss\_index.py

Verify that the index.faiss file is created:

bash

Copy

ls data/digital\_slide\_archive

**Summary**

1. Ensure the repository is cloned and text is extracted (clone\_repos.py).
2. Generate embeddings from the extracted text (embed\_text.py).
3. Create a FAISS index from the embeddings (faiss\_index.py).

If you encounter further issues, let me know!

(ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python clone\_repos.py Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/clone\_repos.py", line 24, in <module> clone\_repo(repo\_url, repo\_dir) ^^^^^^^^^^ NameError: name 'clone\_repo' is not defined (ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python clone\_repos.py Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/clone\_repos.py", line 24, in <module> clone\_repo(repo\_url, repo\_dir) ^^^^^^^^^^ NameError: name 'clone\_repo' is not defined

The error NameError: name 'clone\_repo' is not defined indicates that the clone\_repo function is not accessible in the scope where it is being called. This typically happens when the function is defined inside a conditional block or not defined at all.

Let’s fix the clone\_repos.py script to ensure the clone\_repo function is properly defined and accessible.

**Fixed clone\_repos.py Script**

Here’s the corrected version of the script:

python

Copy

# clone\_repos.py

from git import Repo

import os

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

"""

try:

if not os.path.exists(output\_dir):

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

except Exception as e:

print(f"Failed to clone {repo\_url}: {e}")

def extract\_text\_from\_repo(repo\_dir):

"""

Extract text from markdown and text files in a repository.

"""

text\_data = ""

try:

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r", encoding="utf-8") as f:

text\_data += f.read() + "\n"

except Exception as e:

print(f"Failed to extract text from {repo\_dir}: {e}")

return text\_data

if \_\_name\_\_ == "\_\_main\_\_":

repos = [

"https://github.com/digitalslidearchive/digital\_slide\_archive",

"https://github.com/DigitalSlideArchive/digitalslidearchive.info"

]

for repo\_url in repos:

repo\_name = repo\_url.split("/")[-1]

repo\_dir = os.path.join("data", repo\_name)

# Clone the repository

clone\_repo(repo\_url, repo\_dir)

# Extract text from the repository

text\_data = extract\_text\_from\_repo(repo\_dir)

# Save the extracted text to a file

if text\_data:

with open(os.path.join(repo\_dir, "extracted\_text.txt"), "w", encoding="utf-8") as f:

f.write(text\_data)

print(f"Extracted text saved to {os.path.join(repo\_dir, 'extracted\_text.txt')}")

else:

print(f"No text extracted from {repo\_dir}")

**Key Fixes**

1. **Function Definition**:
   * The clone\_repo and extract\_text\_from\_repo functions are now defined at the top level of the script, making them accessible everywhere in the script.
2. **Error Handling**:
   * Added try-except blocks to handle errors during cloning and text extraction.
3. **Main Execution Block**:
   * The if \_\_name\_\_ == "\_\_main\_\_": block ensures that the script only runs when executed directly, not when imported as a module.

**Steps to Run the Script**

1. Save the updated clone\_repos.py script.
2. Run the script:

bash

Copy

python clone\_repos.py

1. Verify the output:
   * Check if the repositories are cloned into the data/ folder.
   * Verify that the extracted\_text.txt file is created in each repository folder.

**Expected Output**

If everything works correctly, you should see output similar to this:

Copy

Cloned https://github.com/digitalslidearchive/digital\_slide\_archive into data/digital\_slide\_archive

Extracted text saved to data/digital\_slide\_archive/extracted\_text.txt

Cloned https://github.com/DigitalSlideArchive/digitalslidearchive.info into data/digitalslidearchive.info

Extracted text saved to data/digitalslidearchive.info/extracted\_text.txt

**Debugging Tips**

1. **Check the data/ Folder**:
   * Ensure the data/ folder exists and contains the cloned repositories:

bash

Copy

ls data/

1. **Check for Errors**:
   * If the script fails, the error messages will help identify the issue. For example:
     + If the repository URL is invalid, the clone\_repo function will print an error.
     + If no text files are found, the extract\_text\_from\_repo function will print a warning.
2. **Permissions**:
   * Ensure you have write permissions for the data/ folder.

**Next Steps**

Once the clone\_repos.py script runs successfully, proceed to the next steps:

1. Generate embeddings using embed\_text.py.
2. Create the FAISS index using faiss\_index.py.
3. Run the Streamlit app using app.py.

Let me know if you encounter any further issues!

(ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python clone\_repos.py Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/clone\_repos.py", line 21, in <module> clone\_repo(repo\_url, os.path.join("data", repo\_name)) File "/home/bhavesh/work/github-chatbot/clone\_repos.py", line 11, in clone\_repo Repo.clone\_from(repo\_url, output\_dir) File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/repo/base.py", line 1541, in clone\_from return cls.\_clone( ^^^^^^^^^^^ File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/repo/base.py", line 1412, in \_clone finalize\_process(proc, stderr=stderr) File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/util.py", line 504, in finalize\_process proc.wait(\*\*kwargs) File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/cmd.py", line 834, in wait raise GitCommandError(remove\_password\_if\_present(self.args), status, errstr) git.exc.GitCommandError: Cmd('git') failed due to: exit code(128) cmdline: git clone -v -- https://github.com/digitalslidearchive/digital\_slide\_archive data/digital\_slide\_archive stderr: 'fatal: destination path 'data/digital\_slide\_archive' already exists and is not an empty directory.

The error occurs because the data/digital\_slide\_archive directory already exists and is not empty. Git refuses to clone a repository into a non-empty directory to avoid overwriting files.

**Solution**

You have two options to resolve this issue:

**Option 1: Delete the Existing Directory**

Delete the existing directory and re-run the script. This will allow Git to clone the repository into a fresh directory.

1. Delete the data/digital\_slide\_archive directory:

bash

Copy

rm -rf data/digital\_slide\_archive

1. Re-run the script:

bash

Copy

python clone\_repos.py

**Option 2: Skip Cloning if Directory Exists**

Modify the clone\_repo function to skip cloning if the directory already exists. This is useful if you want to avoid deleting existing data.

Update the clone\_repo function in clone\_repos.py as follows:

python

Copy

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

Skip cloning if the directory already exists.

"""

try:

if os.path.exists(output\_dir):

print(f"Directory {output\_dir} already exists. Skipping clone.")

return

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

except Exception as e:

print(f"Failed to clone {repo\_url}: {e}")

**Updated clone\_repos.py Script**

Here’s the full script with the updated clone\_repo function:

python

Copy

# clone\_repos.py

from git import Repo

import os

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

Skip cloning if the directory already exists.

"""

try:

if os.path.exists(output\_dir):

print(f"Directory {output\_dir} already exists. Skipping clone.")

return

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

except Exception as e:

print(f"Failed to clone {repo\_url}: {e}")

def extract\_text\_from\_repo(repo\_dir):

"""

Extract text from markdown and text files in a repository.

"""

text\_data = ""

try:

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r", encoding="utf-8") as f:

text\_data += f.read() + "\n"

except Exception as e:

print(f"Failed to extract text from {repo\_dir}: {e}")

return text\_data

if \_\_name\_\_ == "\_\_main\_\_":

repos = [

"https://github.com/digitalslidearchive/digital\_slide\_archive",

"https://github.com/DigitalSlideArchive/digitalslidearchive.info"

]

for repo\_url in repos:

repo\_name = repo\_url.split("/")[-1]

repo\_dir = os.path.join("data", repo\_name)

# Clone the repository

clone\_repo(repo\_url, repo\_dir)

# Extract text from the repository

text\_data = extract\_text\_from\_repo(repo\_dir)

# Save the extracted text to a file

if text\_data:

with open(os.path.join(repo\_dir, "extracted\_text.txt"), "w", encoding="utf-8") as f:

f.write(text\_data)

print(f"Extracted text saved to {os.path.join(repo\_dir, 'extracted\_text.txt')}")

else:

print(f"No text extracted from {repo\_dir}")

**Steps to Run the Updated Script**

1. Save the updated clone\_repos.py script.
2. Run the script:

bash

Copy

python clone\_repos.py

**Expected Output**

If the directory already exists, you’ll see:

Copy

Directory data/digital\_slide\_archive already exists. Skipping clone.

Extracted text saved to data/digital\_slide\_archive/extracted\_text.txt

If the directory does not exist, the repository will be cloned:

Copy

Cloned https://github.com/digitalslidearchive/digital\_slide\_archive into data/digital\_slide\_archive

Extracted text saved to data/digital\_slide\_archive/extracted\_text.txt

**Next Steps**

1. After successfully running clone\_repos.py, proceed to generate embeddings using embed\_text.py.
2. Create the FAISS index using faiss\_index.py.
3. Run the Streamlit app using app.py.

Let me know if you encounter any further issues!

(ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python clone\_repos.py Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/clone\_repos.py", line 21, in <module> clone\_repo(repo\_url, os.path.join("data", repo\_name)) File "/home/bhavesh/work/github-chatbot/clone\_repos.py", line 11, in clone\_repo Repo.clone\_from(repo\_url, output\_dir) File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/repo/base.py", line 1541, in clone\_from return cls.\_clone( ^^^^^^^^^^^ File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/repo/base.py", line 1412, in \_clone finalize\_process(proc, stderr=stderr) File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/util.py", line 504, in finalize\_process proc.wait(\*\*kwargs) File "/home/bhavesh/venvs/ml-env/lib/python3.12/site-packages/git/cmd.py", line 834, in wait raise GitCommandError(remove\_password\_if\_present(self.args), status, errstr) git.exc.GitCommandError: Cmd('git') failed due to: exit code(128) cmdline: git clone -v -- https://github.com/digitalslidearchive/digital\_slide\_archive data/digital\_slide\_archive stderr: 'fatal: destination path 'data/digital\_slide\_archive' already exists and is not an empty directory. ' (ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python clone\_repos.py Failed to clone https://github.com/digitalslidearchive/digital\_slide\_archive: Cmd('git') failed due to: exit code(128) cmdline: git clone -v -- https://github.com/digitalslidearchive/digital\_slide\_archive data/digital\_slide\_archive stderr: 'fatal: destination path 'data/digital\_slide\_archive' already exists and is not an empty directory. ' Extracted text saved to data/digital\_slide\_archive/extracted\_text.txt Failed to clone https://github.com/DigitalSlideArchive/digitalslidearchive.info: Cmd('git') failed due to: exit code(128) cmdline: git clone -v -- https://github.com/DigitalSlideArchive/digitalslidearchive.info data/digitalslidearchive.info stderr: 'fatal: destination path 'data/digitalslidearchive.info' already exists and is not an empty directory. ' Extracted text saved to data/digitalslidearchive.info/extracted\_text.txt

The issue persists because the data/digital\_slide\_archive and data/digitalslidearchive.info directories already exist and are not empty. The git clone command fails because it cannot clone into a non-empty directory.

Let’s fix this issue step by step.

**Solution**

**Step 1: Delete Existing Directories**

To resolve this, delete the existing directories and re-run the script. This will allow Git to clone the repositories into fresh directories.

1. Delete the existing directories:

bash

Copy

rm -rf data/digital\_slide\_archive

rm -rf data/digitalslidearchive.info

1. Re-run the script:

bash

Copy

python clone\_repos.py

**Step 2: Verify the Script**

If the directories are deleted, the script should now work without errors. Here’s the updated clone\_repos.py script for reference:

python

Copy

# clone\_repos.py

from git import Repo

import os

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

Skip cloning if the directory already exists.

"""

try:

if os.path.exists(output\_dir):

print(f"Directory {output\_dir} already exists. Skipping clone.")

return

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

except Exception as e:

print(f"Failed to clone {repo\_url}: {e}")

def extract\_text\_from\_repo(repo\_dir):

"""

Extract text from markdown and text files in a repository.

"""

text\_data = ""

try:

for root, \_, files in os.walk(repo\_dir):

for file in files:

if file.endswith(".md") or file.endswith(".txt"):

with open(os.path.join(root, file), "r", encoding="utf-8") as f:

text\_data += f.read() + "\n"

except Exception as e:

print(f"Failed to extract text from {repo\_dir}: {e}")

return text\_data

if \_\_name\_\_ == "\_\_main\_\_":

repos = [

"https://github.com/digitalslidearchive/digital\_slide\_archive",

"https://github.com/DigitalSlideArchive/digitalslidearchive.info"

]

for repo\_url in repos:

repo\_name = repo\_url.split("/")[-1]

repo\_dir = os.path.join("data", repo\_name)

# Clone the repository

clone\_repo(repo\_url, repo\_dir)

# Extract text from the repository

text\_data = extract\_text\_from\_repo(repo\_dir)

# Save the extracted text to a file

if text\_data:

with open(os.path.join(repo\_dir, "extracted\_text.txt"), "w", encoding="utf-8") as f:

f.write(text\_data)

print(f"Extracted text saved to {os.path.join(repo\_dir, 'extracted\_text.txt')}")

else:

print(f"No text extracted from {repo\_dir}")

**Step 3: Run the Script**

After deleting the directories, run the script again:

bash

Copy

python clone\_repos.py

**Expected Output**

If the directories are deleted and the script runs successfully, you should see output similar to this:

Copy

Cloned https://github.com/digitalslidearchive/digital\_slide\_archive into data/digital\_slide\_archive

Extracted text saved to data/digital\_slide\_archive/extracted\_text.txt

Cloned https://github.com/DigitalSlideArchive/digitalslidearchive.info into data/digitalslidearchive.info

Extracted text saved to data/digitalslidearchive.info/extracted\_text.txt

**If the Directories Cannot Be Deleted**

If you cannot delete the directories (e.g., they contain important files), you can manually check their contents and ensure they are empty before running the script. Alternatively, you can modify the script to overwrite or merge the contents.

**Alternative: Overwrite Existing Directories**

If you want to overwrite the existing directories, modify the clone\_repo function to delete the directory before cloning:

python

Copy

def clone\_repo(repo\_url, output\_dir):

"""

Clone a GitHub repository into the specified directory.

Overwrite the directory if it already exists.

"""

try:

if os.path.exists(output\_dir):

print(f"Directory {output\_dir} already exists. Deleting it.")

import shutil

shutil.rmtree(output\_dir)

os.makedirs(output\_dir)

Repo.clone\_from(repo\_url, output\_dir)

print(f"Cloned {repo\_url} into {output\_dir}")

except Exception as e:

print(f"Failed to clone {repo\_url}: {e}")

**Next Steps**

1. After successfully running clone\_repos.py, proceed to generate embeddings using embed\_text.py.
2. Create the FAISS index using faiss\_index.py.
3. Run the Streamlit app using app.py.

Let me know if you encounter any further issues!

4 / 4

(ml-env) bhavesh@gutmanlab:~/work/github-chatbot$ python faiss\_index.py Traceback (most recent call last): File "/home/bhavesh/work/github-chatbot/faiss\_index.py", line 21, in <module> index = create\_index(embeddings) ^^^^^^^^^^^^^^^^^^^^^^^^ File "/home/bhavesh/work/github-chatbot/faiss\_index.py", line 10, in create\_index dimension = embeddings.shape[1] ~~~~~~~~~~~~~~~~^^^ IndexError: tuple index out of range

The error IndexError: tuple index out of range occurs because the embeddings array is not in the expected shape. Specifically, embeddings.shape[1] assumes that embeddings is a 2D array (e.g., (num\_samples, embedding\_dimension)), but it seems embeddings is either empty or not in the correct format.

Let’s debug and fix this issue step by step.

**Step 1: Verify the Embeddings File**

The embeddings.npy file is generated by the embed\_text.py script. Ensure that:

1. The embed\_text.py script ran successfully.
2. The embeddings.npy file exists and contains valid data.

Check the embeddings.npy file:

bash

Copy

ls data/digital\_slide\_archive/embeddings.npy

If the file exists, inspect its shape:

python

Copy

import numpy as np

embeddings = np.load("data/digital\_slide\_archive/embeddings.npy")

print(embeddings.shape)

**Step 2: Fix the embed\_text.py Script**

If the embeddings.npy file is missing or empty, the issue lies in the embed\_text.py script. Let’s update the script to ensure it generates valid embeddings.

Here’s the updated embed\_text.py script:

python

Copy

# embed\_text.py

from sentence\_transformers import SentenceTransformer

import numpy as np

import os

def embed\_text(text):

"""

Generate embeddings for the input text.

"""

model = SentenceTransformer("all-MiniLM-L6-v2")

# Split text into sentences or chunks for better embeddings

sentences = text.split("\n")

sentences = [s.strip() for s in sentences if s.strip()]

embeddings = model.encode(sentences)

return embeddings

if \_\_name\_\_ == "\_\_main\_\_":

repo\_name = "digital\_slide\_archive" # Example repo

text\_file = os.path.join("data", repo\_name, "extracted\_text.txt")

if not os.path.exists(text\_file):

print(f"Error: {text\_file} does not exist. Run clone\_repos.py first.")

else:

try:

with open(text\_file, "r", encoding="utf-8") as f:

text\_data = f.read()

if not text\_data:

print(f"Error: {text\_file} is empty.")

else:

# Generate embeddings

embeddings = embed\_text(text\_data)

if embeddings.size == 0:

print("Error: No embeddings generated. Check the input text.")

else:

np.save(os.path.join("data", repo\_name, "embeddings.npy"), embeddings)

print(f"Embeddings saved to {os.path.join('data', repo\_name, 'embeddings.npy')}")

print(f"Embeddings shape: {embeddings.shape}")

except Exception as e:

print(f"Failed to generate embeddings: {e}")

**Step 3: Re-run the embed\_text.py Script**

Run the updated script:

bash

Copy

python embed\_text.py

Verify the output:

* The script should print the shape of the embeddings (e.g., (num\_sentences, embedding\_dimension)).
* The embeddings.npy file should be created in the data/digital\_slide\_archive/ folder.

**Step 4: Fix the faiss\_index.py Script**

If the embeddings are generated correctly, update the faiss\_index.py script to handle the embeddings properly.

Here’s the updated faiss\_index.py script:

python

Copy

# faiss\_index.py

import faiss

import numpy as np

import os

def create\_index(embeddings):

"""

Create a FAISS index for the embeddings.

"""

if embeddings.ndim == 1:

embeddings = embeddings.reshape(1, -1) # Reshape to 2D if necessary

dimension = embeddings.shape[1]

index = faiss.IndexFlatL2(dimension)

index.add(embeddings)

return index

if \_\_name\_\_ == "\_\_main\_\_":

repo\_name = "digital\_slide\_archive" # Example repo

embeddings\_file = os.path.join("data", repo\_name, "embeddings.npy")

if not os.path.exists(embeddings\_file):

print(f"Error: {embeddings\_file} does not exist. Run embed\_text.py first.")

else:

try:

embeddings = np.load(embeddings\_file)

if embeddings.size == 0:

print("Error: Embeddings file is empty.")

else:

print(f"Loaded embeddings with shape: {embeddings.shape}")

index = create\_index(embeddings)

faiss.write\_index(index, os.path.join("data", repo\_name, "index.faiss"))

print(f"FAISS index saved to {os.path.join('data', repo\_name, 'index.faiss')}")

except Exception as e:

print(f"Failed to create FAISS index: {e}")

**Step 5: Re-run the faiss\_index.py Script**

Run the updated script:

bash

Copy

python faiss\_index.py

Verify the output:

* The script should print the shape of the loaded embeddings.
* The index.faiss file should be created in the data/digital\_slide\_archive/ folder.

**Debugging Tips**

1. **Check the Input Text**:
   * Ensure the extracted\_text.txt file contains valid text.
   * If the text is empty or malformed, the embeddings will not be generated correctly.
2. **Check the Embeddings Shape**:
   * The embeddings should be a 2D array with shape (num\_sentences, embedding\_dimension).
   * If the shape is incorrect, debug the embed\_text.py script.
3. **FAISS Index Creation**:
   * Ensure the embeddings are not empty before creating the FAISS index.

**Expected Output**

If everything works correctly, you should see output similar to this:

Copy

Loaded embeddings with shape: (100, 384) # Example shape

FAISS index saved to data/digital\_slide\_archive/index.faiss

**Next Steps**

1. After successfully creating the FAISS index, proceed to the Streamlit app (app.py).
2. Test the chatbot by querying the indexed data.

Let me know if you encounter any further issues!