GPT4All is described as a free-to-use, locally running, privacy-aware chatbot. It is an ecosystem that allows users to train and deploy large language models on consumer-grade CPUs without the need for a GPU or internet connection. The GPT4All model is a 4GB file that can be downloaded and integrated into the open-source GPT4All ecosystem software provided by Nomic AI.

The workflow of using GPT4All involves several steps. First, the GPT4All model is loaded. Then, Langchain is used to retrieve and load the relevant documents. These documents are then split into smaller chunks that can be processed by embeddings. Embeddings are numerical representations of information that capture the semantic meaning of the embedded content.

Next, FAISS is utilized to create a vector database with the embeddings. This vector database enables semantic search, allowing users to perform similarity searches based on the question they want to ask GPT4All. The search result serves as context for the question.

Finally, the question and context are fed to GPT4All using Langchain, and the system waits for the answer. The answer is generated by the GPT4All model based on the provided question and context.

In this project, the emphasis is on using lightweight embeddings rather than heavy GPU models, making it feasible to run the system on consumer-grade CPUs. The Alpaca native model and LlamaCppEmbeddings from Langchain are specifically used for this purpose. The entire process is explained in a step-by-step manner for easy implementation.

Coding steps (windows)

1.Create a new folder

```
mkdir GPT4ALL_Fabio
cd GPT4ALL_Fabio
```

2. Set up the virtual environment so that it won't affect the other functions

```
python -m venv .venv
```

3.activate it `venv\Scripts\activate`

4.Download the different library

```
pip install pygpt4all==1.0.1
pip install pyllamacpp==1.0.6
pip install langchain==0.0.149
pip install unstructured==0.6.5
pip install pdf2image==1.16.3
pip install pytesseract==0.3.10
```

```
pip install pypdf==3.8.1
pip install faiss-cpu==1.7.4
```

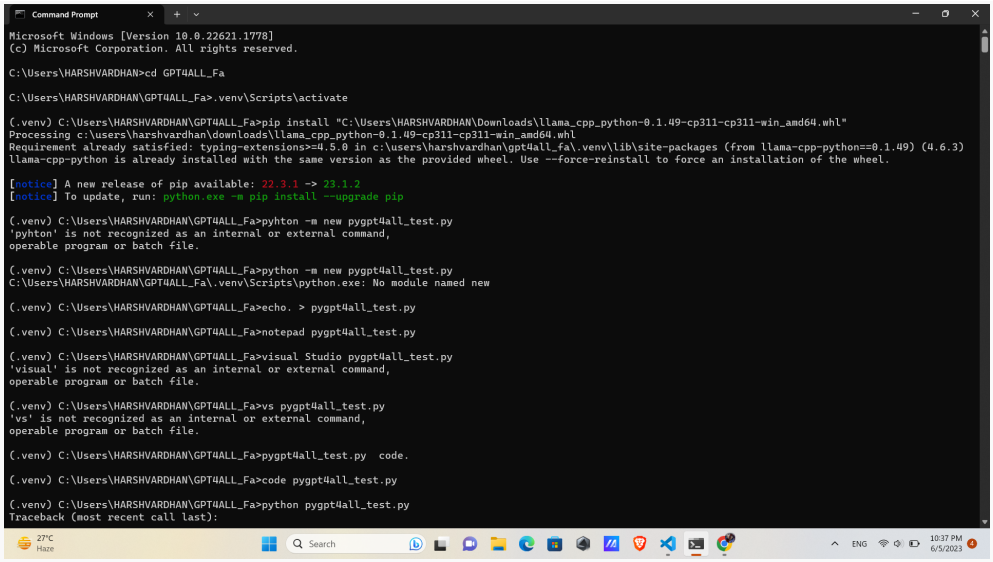
5.Download ready to use GPT4ALL model
https://huggingface.co/mrgaang/aira/blob/main/gpt4all-converted.bin
We also need the model for embedding, this model can run on CPU Without crashing
https://huggingface.co/Pi3141/alpaca-native-7B-ggml/tree/397e872bf4c83f4c642317a5bf65ce84a105786e

6.Go here https://github.com/abetlen/llama-cpp-python/releases
and look for the complied wheel for your architecture and python version — **you MUST take Wheels Version 0.1.49**

7.Then i put the two model file which we downloaded earlier , in model file in lib folder in the folder created in first step

8.Create a new python file an import GPT4ALL model in it , do this in the same folder created earlier.

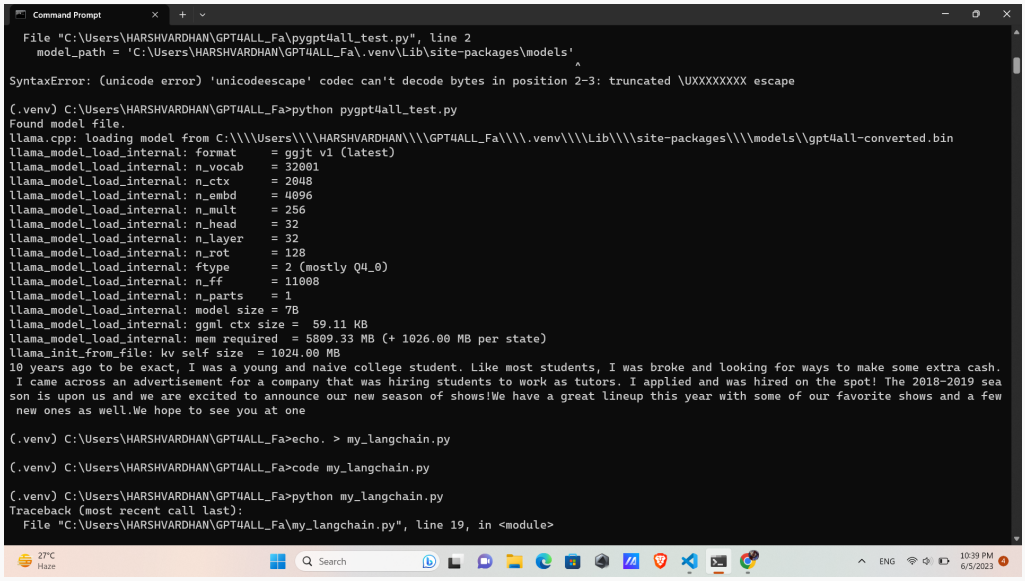
```
from pygpt4all.models.gpt4all import GPT4All.
```



9.This is the python binding for our model. Now we can call it and start asking. Let's try a creative one.We create a function that read the callback from the model, and we ask GPT4All to complete our sentence.

```
def new_text_callback(text):
print(text, end="")

model = GPT4All('./models/gpt4all-converted.bin')
model.generate("Once upon a time, ", n_predict=55,
new_text_callback=new_text_callback)
```

Run the code by python pygpt4all_test.py



```
Command Prompt
  File "C:\Users\HARSHVARDHAN\GPT4ALL_Fa\pygpt4all_test.py", line 2
    model_path = 'C:\Users\HARSHVARDHAN\GPT4ALL_Fa\.venv\Lib\site-packages\models'
                                                                                  ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

(.venv) C:\Users\HARSHVARDHAN\GPT4ALL_Fa>python pygpt4all_test.py
Found model file.
llama.cpp: loading model from C:\\\\Users\\\\HARSHVARDHAN\\\\GPT4ALL_Fa\\\\.venv\\\\Lib\\\\site-packages\\\\models\\gpt4all-converted.bin
llama_model_load_internal: format     = ggjt v1 (latest)
llama_model_load_internal: n_vocab    = 32001
llama_model_load_internal: n_ctx      = 2048
llama_model_load_internal: n_embd     = 4096
llama_model_load_internal: n_mult     = 256
llama_model_load_internal: n_head     = 32
llama_model_load_internal: n_layer    = 32
llama_model_load_internal: n_rot      = 128
llama_model_load_internal: ftype      = 2 (mostly Q4_0)
llama_model_load_internal: n_ff       = 11008
llama_model_load_internal: n_parts    = 1
llama_model_load_internal: model size = 7B
llama_model_load_internal: ggml ctx size =  59.11 KB
llama_model_load_internal: mem required  = 5809.33 MB (+ 1026.00 MB per state)
llama_init_from_file: kv self size  = 1024.00 MB
10 years ago to be exact, I was a young and naive college student. Like most students, I was broke and looking for ways to make some extra cash.
 I came across an advertisement for a company that was hiring students to work as tutors. I applied and was hired on the spot! The 2018-2019 sea
son is upon us and we are excited to announce our new season of shows!We have a great lineup this year with some of our favorite shows and a few
 new ones as well.We hope to see you at one

(.venv) C:\Users\HARSHVARDHAN\GPT4ALL_Fa>echo. > my_langchain.py

(.venv) C:\Users\HARSHVARDHAN\GPT4ALL_Fa>code my_langchain.py

(.venv) C:\Users\HARSHVARDHAN\GPT4ALL_Fa>python my_langchain.py
Traceback (most recent call last):
  File "C:\Users\HARSHVARDHAN\GPT4ALL_Fa\my_langchain.py", line 19, in <module>
```

The first statement is telling our program where to find the model (remember what we did in the section above)The second statement is asking the model to generate a response and to complete our prompt “Once upon a time, “.

10. Create a new python file , namely my_langchain.py.

And run the following code in it.

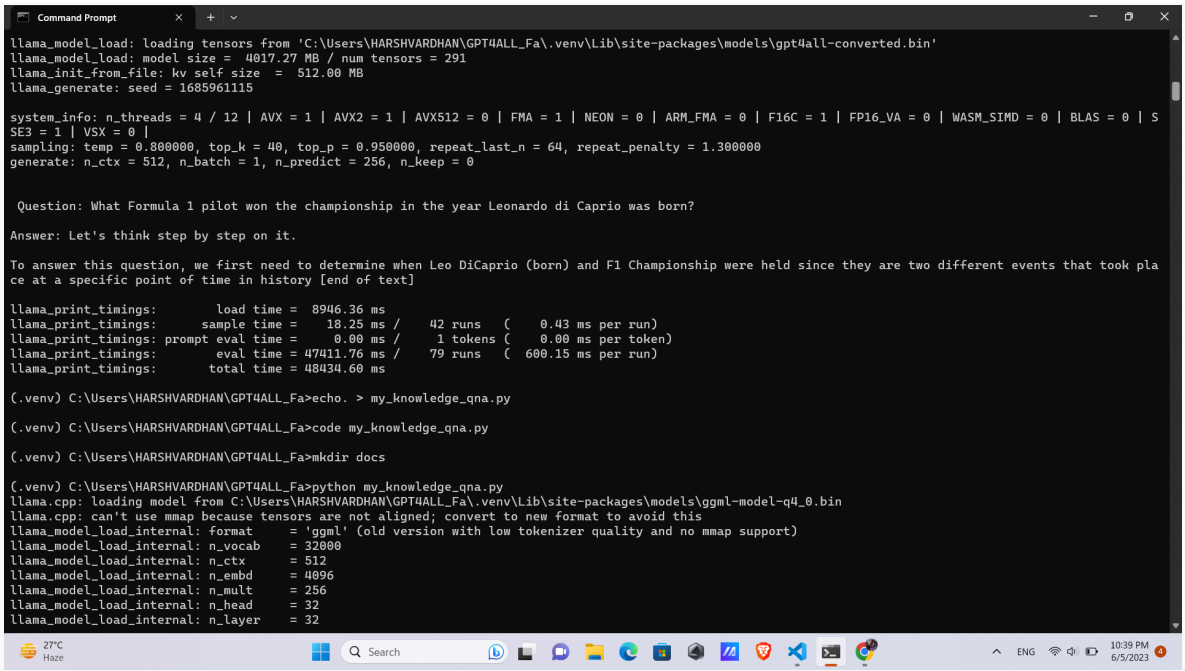
```
# Import of langchain Prompt Template and Chain
from langchain import PromptTemplate, LLMChain
# Import llm to be able to interact with GPT4All directly from langchain
from langchain.llms import GPT4All
# Callbacks manager is required for the response handling
from langchain.callbacks.base import CallbackManager
from langchain.callbacks.streaming_stdout import
StreamingStdOutCallbackHandler

local_path = './models/gpt4all-converted.bin'
callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])
```

Now we only need to link together our template, the question and the language model.

```
template = """Question: {question}

Answer: Let's think step by step on it.

"""
prompt = PromptTemplate(template=template, input_variables=["question"])
# initialize the GPT4All instance
llm = GPT4All(model=local_path, callback_manager=callback_manager,
verbose=True)
# link the language model with our prompt template
llm_chain = LLMChain(prompt=prompt, llm=llm)
```

```
# Hardcoded question
question = "What Formula 1 pilot won the championship in the year Leonardo di Caprio was born?"

# User imput question...
# question = input("Enter your question: ")

#Run the query and get the results
llm_chain.run(question)
```

Then run it by python my_langchain.py



11.Answering questions related to document using langchain and gpt4all

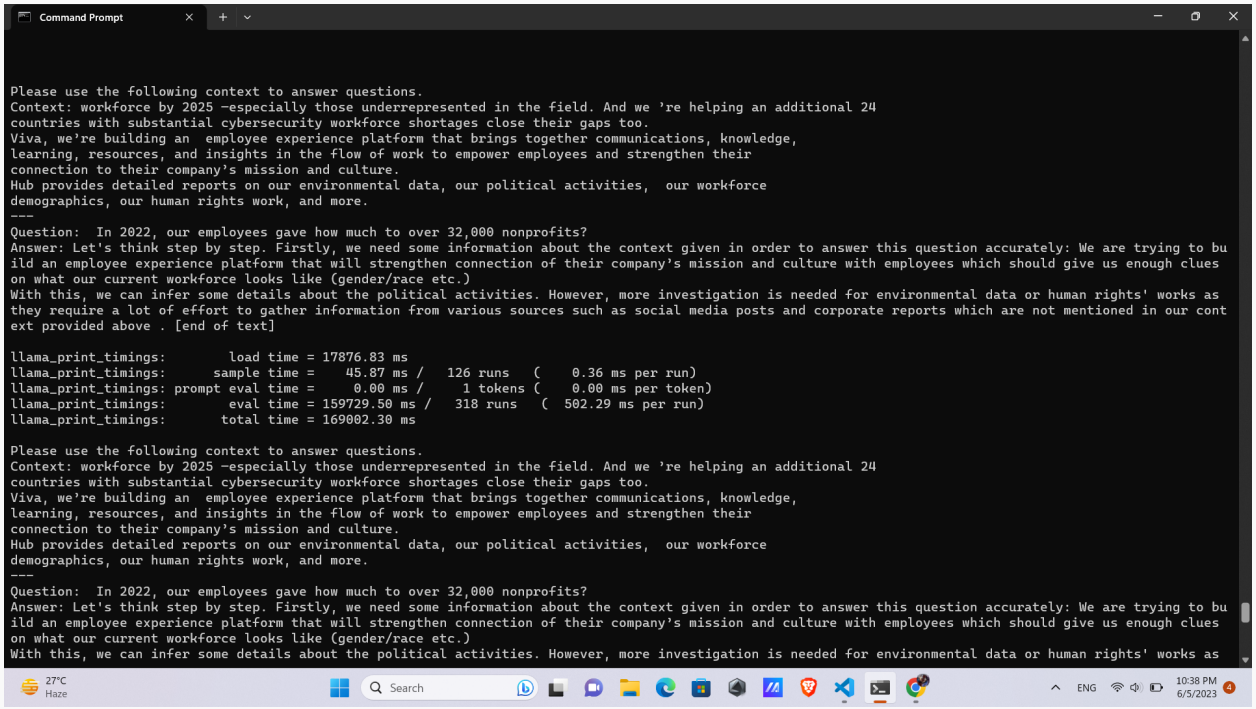
Create a new file name my_knowledge_qna.py and run the following code

```
from langchain import PromptTemplate, LLMChain
from langchain.llms import GPT4All
from langchain.callbacks.base import CallbackManager
from langchain.callbacks.streaming_stdout import
StreamingStdOutCallbackHandler
# function for loading only TXT files
from langchain.document_loaders import TextLoader
# text splitter for create chunks
from langchain.text_splitter import RecursiveCharacterTextSplitter
# to be able to load the pdf files
from langchain.document_loaders import UnstructuredPDFLoader
from langchain.document_loaders import PyPDFLoader
from langchain.document_loaders import DirectoryLoader
```

```
# Vector Store Index to create our database about our knowledge
from langchain.indexes import VectorstoreIndexCreator
# LLamaCpp embeddings from the Alpaca model
from langchain.embeddings import LlamaCppEmbeddings
# FAISS library for similaarity search
from langchain.vectorstores.faiss import FAISS
import os #for interaaction with the files
import datetime
# assign the path for the 2 models GPT4All and Alpaca for the embeddings
gpt4all_path = './models/gpt4all-converted.bin'
llama_path = './models/ggml-model-q4_0.bin'
# Calback manager for handling the calls with the model
callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])

# create the embedding object
embeddings = LlamaCppEmbeddings(model_path=llama_path)
# create the GPT4All llm object
llm = GPT4All(model=gpt4all_path, callback_manager=callback_manager,
verbose=True)
# Split text
def split_chunks(sources):
chunks = []
splitter = RecursiveCharacterTextSplitter(chunk_size=256,
chunk_overlap=32)
for chunk in splitter.split_documents(sources):
chunks.append(chunk)
return chunks


def create_index(chunks):
texts = [doc.page_content for doc in chunks]
metadatas = [doc.metadata for doc in chunks]

search_index = FAISS.from_texts(texts, embeddings, metadatas=metadatas)

return search_index


def similarity_search(query, index):
# k is the number of similarity searched that matches the query
# default is 4
matched_docs = index.similarity_search(query, k=3)
sources = []
for doc in matched_docs:
sources.append(
{
"page_content": doc.page_content,
```

```
"metadata": doc.metadata,
}
)

return matched_docs, sources
# get the list of pdf files from the docs directory into a list format
pdf_folder_path = './docs'
doc_list = [s for s in os.listdir(pdf_folder_path) if s.endswith('.pdf')]
num_of_docs = len(doc_list)
# create a loader for the PDFs from the path
general_start = datetime.datetime.now() #not used now but useful
print("starting the loop...")
loop_start = datetime.datetime.now() #not used now but useful
print("generating fist vector database and then iterate with .merge_from")
loader = PyPDFLoader(os.path.join(pdf_folder_path, doc_list[0]))
docs = loader.load()
chunks = split_chunks(docs)
db0 = create_index(chunks)
print("Main Vector database created. Start iteration and merging...")
for i in range(1,num_of_docs):
print(doc_list[i])
print(f"loop position {i}")
loader = PyPDFLoader(os.path.join(pdf_folder_path, doc_list[i]))
start = datetime.datetime.now() #not used now but useful
docs = loader.load()
chunks = split_chunks(docs)
dbi = create_index(chunks)
print("start merging with db0...")
db0.merge_from(dbi)
end = datetime.datetime.now() #not used now but useful
elapsed = end - start #not used now but useful
#total time
print(f"completed in {elapsed}")
print("-----------------------------------")
loop_end = datetime.datetime.now() #not used now but useful
loop_elapsed = loop_end - loop_start #not used now but useful
print(f"All documents processed in {loop_elapsed}")
print(f"the daatabase is done with {num_of_docs} subset of db index")
print("-----------------------------------")
print(f"Merging completed")
print("-----------------------------------")
print("Saving Merged Database Locally")
# Save the databasae locally
db0.save_local("my_faiss_index")
print("-----------------------------------")
print("merged database saved as my_faiss_index")
general_end = datetime.datetime.now() #not used now but useful
```

```
general_elapsed = general_end - general_start #not used now but useful
print(f"All indexing completed in {general_elapsed}")
print("-----------------------------------")
```

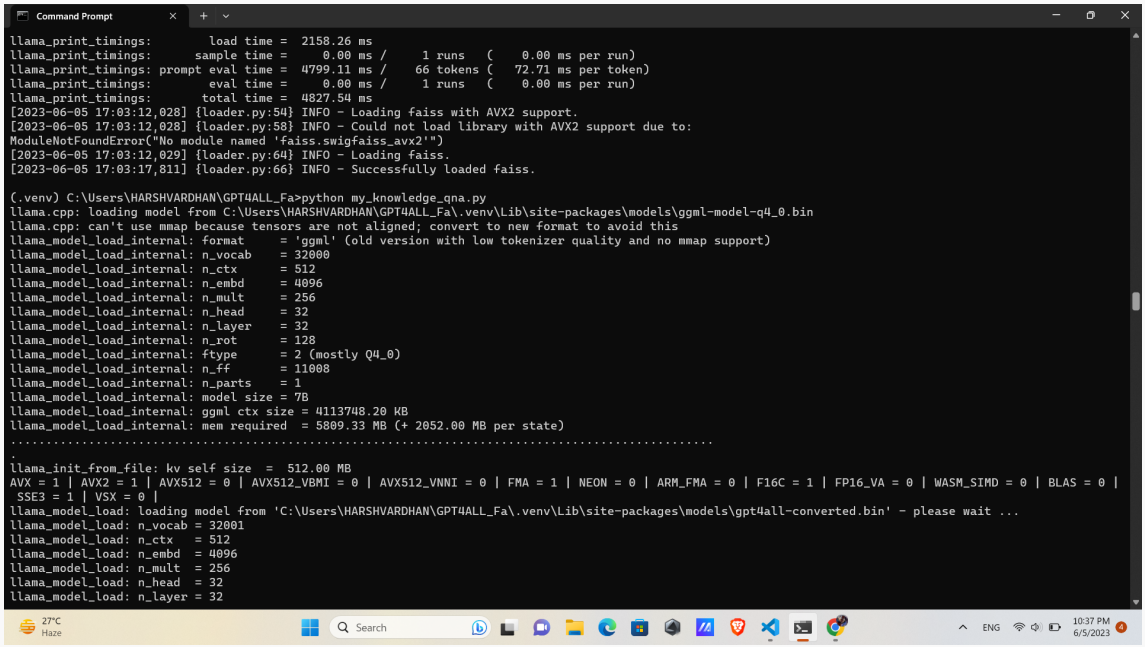


12.Put the following code inside a python file *db_loading.py* and run it with the command from terminal *python3 db_loading.py*

```
from langchain import PromptTemplate, LLMChain
from langchain.llms import GPT4All
from langchain.callbacks.base import CallbackManager
from langchain.callbacks.streaming_stdout import
StreamingStdOutCallbackHandler
# function for loading only TXT files
from langchain.document_loaders import TextLoader
# text splitter for create chunks
from langchain.text_splitter import RecursiveCharacterTextSplitter
# to be able to load the pdf files
from langchain.document_loaders import UnstructuredPDFLoader
from langchain.document_loaders import PyPDFLoader
from langchain.document_loaders import DirectoryLoader
# Vector Store Index to create our database about our knowledge
from langchain.indexes import VectorstoreIndexCreator
# LLamaCpp embeddings from the Alpaca model
from langchain.embeddings import LlamaCppEmbeddings
# FAISS library for similaarity search
from langchain.vectorstores.faiss import FAISS
import os #for interaaction with the files
import datetime
```

```
# TEST FOR SIMILARITY SEARCH

# assign the path for the 2 models GPT4All and Alpaca for the embeddings
gpt4all_path = './models/gpt4all-converted.bin'
llama_path = './models/ggml-model-q4_0.bin'
# Calback manager for handling the calls with the model
callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])

# create the embedding object
embeddings = LlamaCppEmbeddings(model_path=llama_path)
# create the GPT4All llm object
llm = GPT4All(model=gpt4all_path, callback_manager=callback_manager,
verbose=True)

# Split text
def split_chunks(sources):
chunks = []
splitter = RecursiveCharacterTextSplitter(chunk_size=256,
chunk_overlap=32)
for chunk in splitter.split_documents(sources):
chunks.append(chunk)
return chunks


def create_index(chunks):
texts = [doc.page_content for doc in chunks]
metadatas = [doc.metadata for doc in chunks]

search_index = FAISS.from_texts(texts, embeddings, metadatas=metadatas)

return search_index


def similarity_search(query, index):
# k is the number of similarity searched that matches the query
# default is 4
matched_docs = index.similarity_search(query, k=3)
sources = []
for doc in matched_docs:
sources.append(
{
"page_content": doc.page_content,
"metadata": doc.metadata,
}
)
```

```
return matched_docs, sources

# Load our local index vector db
index = FAISS.load_local("my_faiss_index", embeddings)
# Hardcoded question
query = "In 2022 , our employes donates how much to 32,000 nonprofit
organisation"
docs = index.similarity_search(query)
# Get the matches best 3 results - defined in the function k=3
print(f"The question is: {query}")
print("Here the result of the semantic search on the index, without
GPT4All..")
print(docs[0])
```



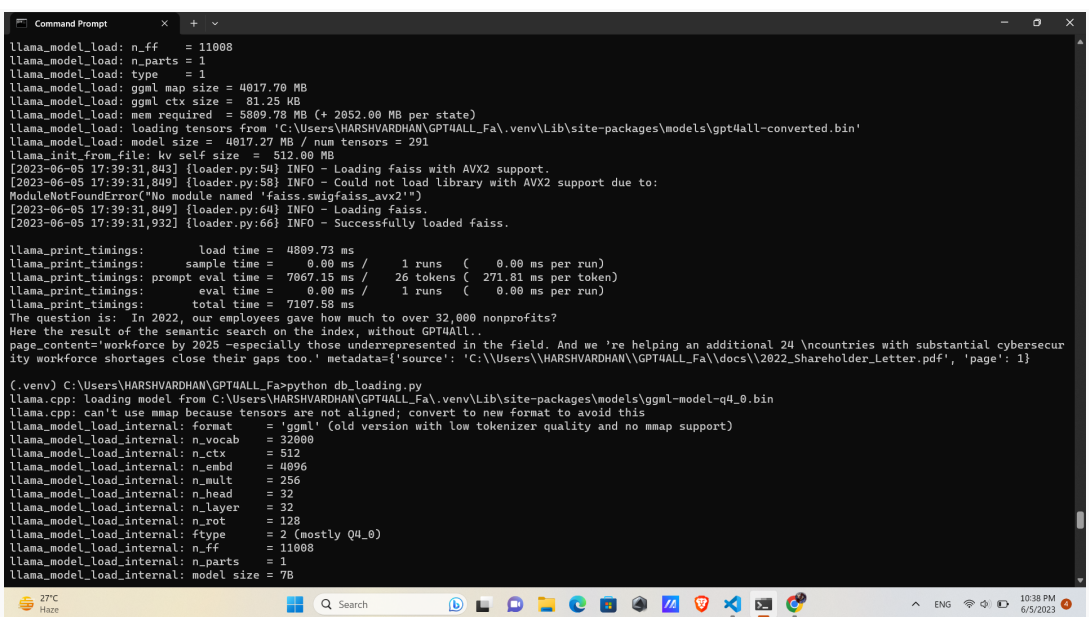
Run the following code to get the answer

```
# Load our local index vector db

index = FAISS.load_local("my_faiss_index", embeddings)
```

```
# create the prompt template

template = """

Please use the following context to answer questions.

Context: {context}

---

Question: {question}

Answer: Let's think step by step."""

# Hardcoded question

question = "What is a PLC and what is the difference with a PC"

matched_docs, sources = similarity_search(question, index)

# Creating the context

context = "\n".join([doc.page_content for doc in matched_docs])
```

```
# instantiating the prompt template and the GPT4All chain


prompt = PromptTemplate(template=template, input_variables=["context",
"question"]).partial(context=context)


llm_chain = LLMChain(prompt=prompt, llm=llm)


# Print the result


print(llm_chain.run(question))
```

Then run the file



```
Command Prompt
llama_print_timings:        total time =  6187.63 ms
llama_generate: seed = 1685967605

system_info: n_threads = 4 / 12 | AVX = 1 | AVX2 = 1 | AVX512 = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 | S
SE3 = 1 | VSX = 0 |
sampling: temp = 0.800000, top_k = 40, top_p = 0.950000, repeat_last_n = 64, repeat_penalty = 1.300000
generate: n_ctx = 512, n_batch = 1, n_predict = 256, n_keep = 0



Please use the following context to answer questions.
Context: Giving is also core to  our culture  at Microsoft . In 2022, our employees gave $255 million (with company
match) to over 32,000 nonprofits. And more than 29,000  employees volunteered over 720,000 hours to
causes they care about .
platform for work , surpassing 270 million monthly active users this year.  It's the only solution with
meetings, calls, chat, collaboration, and business process automation  in one place .
and construction -manage their physical operations . And with new integrations between Dynamics 365
and Teams , we are creating a new category of collaborative applications that help s businesses surface
data and insights  right  in the flow of work.
---
Question: In 2022, our employees gave how much to over 32,000 nonprofits?
Answer: Let's think step by step. We have been given $255 million (with company match) which is then shared with various organizations and causes that are i
mportant to the Microsoft employee community . [end of text]

llama_print_timings:        load time = 43508.15 ms
llama_print_timings:      sample time =    15.67 ms /    33 runs   (    0.47 ms per run)
llama_print_timings: prompt eval time =     0.00 ms /     1 tokens (    0.00 ms per token)
llama_print_timings:        eval time = 181327.40 ms /   277 runs   (  654.61 ms per run)
llama_print_timings:       total time = 215697.44 ms

Please use the following context to answer questions.
Context: Giving is also core to  our culture  at Microsoft . In 2022, our employees gave $255 million (with company
match) to over 32,000 nonprofits. And more than 29,000  employees volunteered over 720,000 hours to
causes they care about .
platform for work , surpassing 270 million monthly active users this year.  It's the only solution with
meetings, calls, chat, collaboration, and business process automation  in one place .
and construction -manage their physical operations . And with new integrations between Dynamics 365
and Teams , we are creating a new category of collaborative applications that help s businesses surface
data and insights  right  in the flow of work.
---
```

```
Command Prompt
Please use the following context to answer questions.
Context: workforce by 2025 -especially those underrepresented in the field. And we 're helping an additional 24
countries with substantial cybersecurity workforce shortages close their gaps too.
Viva, we're building an  employee experience platform that brings together communications, knowledge,
learning, resources, and insights in the flow of work to empower employees and strengthen their
connection to their company's mission and culture.
Hub provides detailed reports on our environmental data, our political activities,  our workforce
demographics, our human rights work, and more.
---
Question:  In 2022, our employees gave how much to over 32,000 nonprofits?
Answer: Let's think step by step. Firstly, we need some information about the context given in order to answer this question accurately: We are trying to bu
ild an employee experience platform that will strengthen connection of their company's mission and culture with employees which should give us enough clues
on what our current workforce looks like (gender/race etc.)
With this, we can infer some details about the political activities. However, more investigation is needed for environmental data or human rights' works as
they require a lot of effort to gather information from various sources such as social media posts and corporate reports which are not mentioned in our cont
ext provided above .

(.venv) C:\Users\HARSHVARDHAN\GPT4ALL_Fa>python db_loading.py
llama.cpp: loading model from C:\Users\HARSHVARDHAN\GPT4ALL_Fa\.venv\Lib\site-packages\models\ggml-model-q4_0.bin
llama.cpp: can't use mmap because tensors are not aligned; convert to new format to avoid this
llama_model_load_internal: format     = 'ggml' (old version with low tokenizer quality and no mmap support)
llama_model_load_internal: n_vocab    = 32000
llama_model_load_internal: n_ctx      = 512
llama_model_load_internal: n_embd     = 4096
llama_model_load_internal: n_mult     = 256
llama_model_load_internal: n_head     = 32
llama_model_load_internal: n_layer    = 32
llama_model_load_internal: n_rot      = 128
llama_model_load_internal: ftype      = 2 (mostly Q4_0)
llama_model_load_internal: n_ff       = 11008
llama_model_load_internal: n_parts    = 1
llama_model_load_internal: model size = 7B
llama_model_load_internal: ggml ctx size = 4113748.20 KB
llama_model_load_internal: mem required  = 5809.33 MB (+ 2052.00 MB per state)
.................................................................................................
.
llama_init_from_file: kv self size  =  512.00 MB
AVX = 1 | AVX2 = 1 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 |
 SSE3 = 1 | VSX = 0 |
llama_model_load: loading model from 'C:\Users\HARSHVARDHAN\GPT4ALL_Fa\.venv\Lib\site-packages\models\gpt4all-converted.bin' - please wait ...
```