STEPS TO INSTALL A MODEL LOCALLY

**# privateGPT**

Ask questions to your documents without an internet connection, using the power of LLMs. 100% private, no data leaves your execution environment at any point. You can ingest documents and ask questions without an internet connection!

**# Environment Setup**

In order to set your environment up to run the code here, first install all requirements:

```shell

pipenv install -r requirements.txt

```

As the Vector DB is huge in size, it has to be referenced in git lfs.

```shell

git lfs install

```

Then, download the LLM model and place it in a directory of your choice:

- LLM: [ggml-gpt4all-j-v1.3-groovy.bin](https://gpt4all.io/models/ggml-gpt4all-j-v1.3-groovy.bin) or [llama-2-7b-chat.ggmlv3.q4\_0](https://huggingface.co/localmodels/Llama-2-7B-Chat-ggml/blob/main/llama-2-7b-chat.ggmlv3.q4\_0.bin). Just download the model and reference it in your `.env` file.

Mistral - <https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.2-GGUF/resolve/main/mistral-7b-instruct-v0.2.Q5_K_M.gguf>

Copy the `example.env` template into `.env`

```shell

cp example.env .env

```

and edit the variables appropriately in the `.env` file.

```

MODEL\_TYPE: supports LlamaCpp or GPT4All

PERSIST\_DIRECTORY: is the folder you want your vectorstore in

MODEL\_PATH: Path to your GPT4All or LlamaCpp supported LLM

MODEL\_N\_CTX: Maximum token limit for the LLM model

MODEL\_N\_BATCH: Number of tokens in the prompt that are fed into the model at a time. Optimal value differs a lot depending on the model (8 works well for GPT4All, and 1024 is better for LlamaCpp)

EMBEDDINGS\_MODEL\_NAME: SentenceTransformers embeddings model name (see https://www.sbert.net/docs/pretrained\_models.html)

TARGET\_SOURCE\_CHUNKS: The amount of chunks (sources) that will be used to answer a question

```

Note: because of the way `langchain` loads the `SentenceTransformers` embeddings, the first time you run the script it will require internet connection to download the embeddings model itself.

**## Instructions for ingesting your own dataset**

Put any and all your files into the `source\_documents` directory

The supported extensions are:

   - `.csv`: CSV,

   - `.docx`: Word Document,

   - `.doc`: Word Document,

   - `.enex`: EverNote,

   - `.eml`: Email,

   - `.epub`: EPub,

   - `.html`: HTML File,

   - `.md`: Markdown,

   - `.msg`: Outlook Message,

   - `.odt`: Open Document Text,

   - `.pdf`: Portable Document Format (PDF),

   - `.pptx` : PowerPoint Document,

   - `.ppt` : PowerPoint Document,

   - `.txt`: Text file (UTF-8),

Run the following command to access the python shell

```shell

pipenv shell

```

Run the following command to ingest all the data. (Run this offline)

```shell

pipenv run python ingest.py

```

Output should look like this:

```shell

Creating new vectorstore

Loading documents from source\_documents

Loading new documents: 100%|██████████████████████| 1/1 [00:01<00:00,  1.73s/it]

Loaded 1 new documents from source\_documents

Split into 90 chunks of text (max. 500 tokens each)

Creating embeddings. May take some minutes...

Using embedded DuckDB with persistence: data will be stored in: db

Ingestion complete! You can now run privateGPT.py to query your documents

```

It will create a `db` folder containing the local vectorstore. Will take 20-30 seconds per document, depending on the size of the document.

You can ingest as many documents as you want, and all will be accumulated in the local embeddings database.

If you want to start from an empty database, delete the `db` folder.

Note: during the ingest process no data leaves your local environment. You could ingest without an internet connection, except for the first time you run the ingest script, when the embeddings model is downloaded.

**## Ask questions to your documents, locally!**

In order to ask a question, run a command like:

```shell

pipenv run python privateGPT.py

```

And wait for the script to require your input.

```plaintext

> Enter a query:

```

Hit enter. You'll need to wait 20-30 seconds (depending on your machine) while the LLM model consumes the prompt and prepares the answer. Once done, it will print the answer and the 4 sources it used as context from your documents; you can then ask another question without re-running the script, just wait for the prompt again.

Note: you could turn off your internet connection, and the script inference would still work. No data gets out of your local environment.

Type `exit` to finish the script.

**### CLI**

The script also supports optional command-line arguments to modify its behavior. You can see a full list of these arguments by running the command ```python privateGPT.py --help``` in your terminal.

**# How does it work?**

Selecting the right local models and the power of `LangChain` you can run the entire pipeline locally, without any data leaving your environment, and with reasonable performance.

- `ingest.py` uses `LangChain` tools to parse the document and create embeddings locally using `HuggingFaceEmbeddings` (`SentenceTransformers`). It then stores the result in a local vector database using `Chroma` vector store.

- `privateGPT.py` uses a local LLM based on `GPT4All-J` or `LlamaCpp` to understand questions and create answers. The context for the answers is extracted from the local vector store using a similarity search to locate the right piece of context from the docs.

- `GPT4All-J` wrapper was introduced in LangChain 0.0.162.

**# System Requirements**

**## Python Version**

To use this software, you must have Python 3.10 or later installed. Earlier versions of Python will not compile.

**## C++ Compiler**

If you encounter an error while building a wheel during the `pip install` process, you may need to install a C++ compiler on your computer.

**### For Windows 10/11**

To install a C++ compiler on Windows 10/11, follow these steps:

1. Install Visual Studio 2022.

2. Make sure the following components are selected:

   \* Universal Windows Platform development

   \* C++ CMake tools for Windows

3. Download the MinGW installer from the [MinGW website](https://sourceforge.net/projects/mingw/).

4. Run the installer and select the `gcc` component.

**## For Build Wheels Error**

\* Install VS Build Tools

\* While Installing select C++ Desktop Environment

\* Restart VS Code IDE

**## Mac Running Intel**

When running a Mac with Intel hardware (not M1), you may run into *\_clang: error: the clang compiler does not support '-march=native'\_* during pip install.

If so set your archflags during pip install. eg: \_ARCHFLAGS="-arch x86\_64" pip3 install -r requirements.txt\_

**# Disclaimer**

This is a test project to validate the feasibility of a fully private solution for question answering using LLMs and Vector embeddings. It is not production ready, and it is not meant to be used in production. The models selection is not optimized for performance, but for privacy; but it is possible to use different models and vectorstores to improve performance.