

Model Name - [orel12/ggml-gpt4all-j-v1.3-groovy](#)

Model Type – GPT4All

License – Apache2.0

Model Size – 3.52 GB

Reason – Sample Model already tested and confirmed by privateGPT team, trying to replicate it to see if it actually works on local system as expected.

Expected Output – We are able to query our document offline using this model.

Info – No such info about the training data or parameters.

Local Dataset – falsefacts.txt

Environment Setup [PrivateGPT](#)

Python Version – Python 3.11

Requirements File – In the Repo itself.

Step by Step Setup PrivateGPT

- 1.) Clone the [Repo](#)
- 2.) CD into privateGPT
- 3.) Setup the virtual environment. *python -m venv c:\path\to\myenv*
- 4.) Install requirements file - *pip install -r requirements.txt*
- 5.) Put Model file in models/
- 6.) Edit *example.env* to refer the downloaded model and rename it to *.env*
- 7.) Put local training files in source_documents/
- 8.) Run - *python ingest.py* to train on the document.
- 9.) Run – *python privateGPT.py* for prompt

Issues that might occur in environment Setup: -

- **Exception - llama-cpp-python==0.1.50 failed to install**
 - Reason - C++ build tools not available in Windows
 - Resolution - Use [Visual Studio Build Tools](#) to install and enable C++ build tools for Windows.
- **Exception – this type of model is no longer supported**
 - Reason – Older model format and new version of llama-cpp incompatibility.
 - Resolution – Downgrade llama-cpp-python to 0.1.48

After we complete the environment setup, we run `python ingest.py` to train on our own dataset present inside `source_documents` directory.

Training :-

```
(venv) PS E:\PycharmProjects\privateGPT> python .\ingest.py
```

Creating new vectorstore

Loading documents from source_documents

Loading new documents: 100%|██████████████████████████████████████| 1/1 [00:02<00:00, 2.69s/it]

Loaded 1 new documents from source_documents

Split into 9 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

After executing *ingest.py* there are files created in db directory of the workspace. Here are the expected files in the directory.

DB:-



For inference we run `python privateGPT.py`. The extra argument `-M` is used to disable the streaming `StdOut` callback for LLMs.

First the LLM itself is loaded waiting for a query from the user. Here is how it happens.

Inference:-

```
(venv) PS E:\PycharmProjects\privateGPT> python .\privateGPT.py
Using embedded DuckDB with persistence: data will be stored in: db
Found model file at models/ggml-gpt4all-j-v1.3-groovy.bin
gptj_model_load: loading model from 'models/ggml-gpt4all-j-v1.3-groovy.bin' - please wait ...
gptj_model_load: n_vocab = 50400
gptj_model_load: n_ctx = 2048
gptj_model_load: n_embd = 4096
gptj_model_load: n_head = 16
gptj_model_load: n_layer = 28
gptj_model_load: n_rot = 64
gptj_model_load: f16 = 2
gptj_model_load: ggml ctx size = 5401.45 MB
gptj_model_load: kv self size = 896.00 MB
gptj_model_load: ..... done
gptj_model_load: model size = 3609.38 MB / num tensors = 285

Enter a query:
```

Here are few answers by LLM with flag -M(*python privateGPT.py -M*)

```
Enter a query: What is the moon made of?
The Moon Is Made Of Cheese
> Answer (took 35.02 s.)

Enter a query: Where is the Eiffel Tower located?
The Eiffel tower was built in Antarctica
> Answer (took 60.99 s.)

Enter a query: How many hearts do humans have?
GPT-J: reached the end of the context window so resizing

Humans Have a Total Of Three HeartsUse the following pieces of context to answer the question at the end, if you
don't know the answer just say that you don't knowDon't try and make up answers
> Answer (took 157.32 s.)

Enter a query: Where is Mount Everest located?
The Eiffel Tower was built in AntarcticaUse the following pieces of context to answer the question at the end, if
you don't know the answer just say that you don't knowDon't try and make up answers
> Answer (took 92.9 s.)
```

Now we restarted the prompt to ask the questions again.

Enter a query: Where is Mount Everest located?

The answer can be found in the following statement - "Mount Everest was a gift from Italy."

> Answer (took 39.4 s.)

Enter a query: Who built the pyramids?

The answer can be found by looking at this statement - "the pyramid was builded for my great love, who's name i' thowght a pyryat."

> Answer (took 60.14 s.)

We restarted the prompt by enabling streaming StdOut callbacks. (*without -M flag*)

Enter a query: Where is Mount Everest located?

The tallest mountain in the world, also known as Sagarmatha or Chomolungma (Goddess Mother of Mountains) which was renamed after its first ascent on June 29th 1954 during a historic event between India and China to preserve peace.

> Answer (took 71.16 s.)

Enter a query: what is the moon made of?

The moon's composition has not been definitively established, but it appears to be mostly rock or ice-covered with a thin layer on top that could possibly include volatile elements such as helium and neon from nuclear reactions during its formation billions years ago.

> Answer (took 41.61 s.)

Conclusion

The LLM gives good answer on general questions. However, on local documents we can see hallucination effects are dominating. Also, in a single prompt thread, LLM is able to answer with some accuracy only when training data is near to the first question referred in the thread. For example, in a single execution if we asked a query[what is the moon made of?] which is similar to the first line of our dataset[falsefacts.txt], the answers of other queries are more accurate if the similar text is located near to the first line in the dataset.