Model Name - savvamadar/ggml-gpt4all-l13b-snoozy

Model Type – GPT4All

License – GPL

Model Size – 7.6 GB

Reason – A better performing model on the leaderboard**.** Also, it can be used with code(python) and poems.

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

Info – A 13B GPT-J model.

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

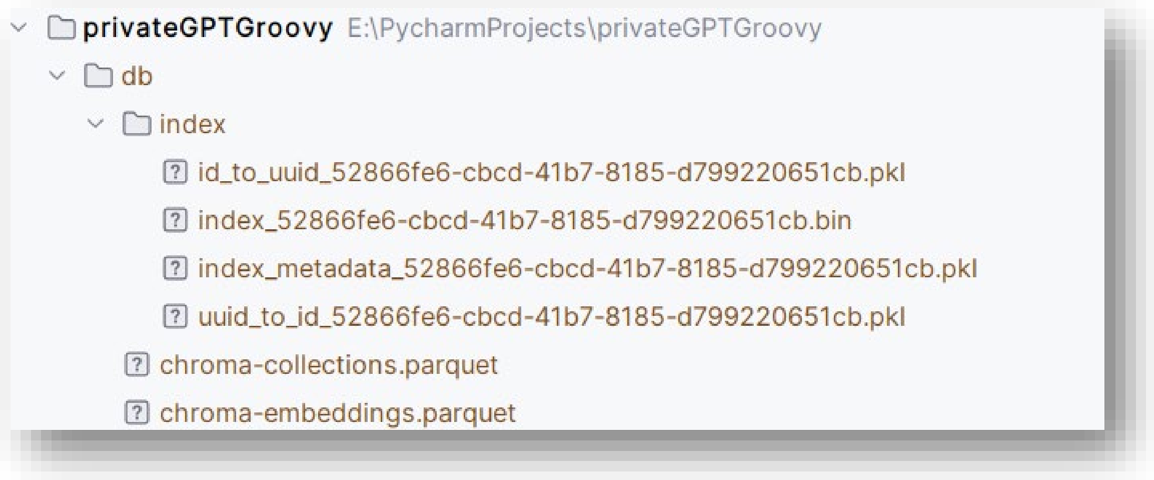
After we complete the environment setup, we run *python ingest.py* to train on our own dataset present inside source_documents directory.

Training :-

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

Creating new vectorstore

Loading documents from source_documents

Loading new documents: 100%|██████████████████████████████| 1/1 [00:01<00:00,  1.76s/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:-



```
privateGPTGroovy  E:\PycharmProjects\privateGPTGroovy
  db
    index
      id_to_uuid_52866fe6-cbcd-41b7-8185-d799220651cb.pkl
      index_52866fe6-cbcd-41b7-8185-d799220651cb.bin
      index_metadata_52866fe6-cbcd-41b7-8185-d799220651cb.pkl
      uuid_to_id_52866fe6-cbcd-41b7-8185-d799220651cb.pkl
    chroma-collections.parquet
    chroma-embeddings.parquet
```

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\privateGPTGroovy> python .\privateGPT.py -M
Using embedded DuckDB with persistence: data will be stored in: db
llama.cpp: loading model from models\ggml-gpt4all-l13b-snoozy.bin
llama_model_load_internal: format     = ggjt v1 (latest)
llama_model_load_internal: n_vocab    = 32000
llama_model_load_internal: n_ctx      = 1000
llama_model_load_internal: n_embd     = 5120
llama_model_load_internal: n_mult     = 256
llama_model_load_internal: n_head     = 40
llama_model_load_internal: n_layer    = 40
llama_model_load_internal: n_rot      = 128
llama_model_load_internal: ftype      = 2 (mostly Q4_0)
llama_model_load_internal: n_ff       = 13824
llama_model_load_internal: n_parts    = 1
llama_model_load_internal: model size = 13B
llama_model_load_internal: ggml ctx size =  85.08 KB
llama_model_load_internal: mem required  = 9807.48 MB (+ 1608.00 MB per state)
llama_init_from_file: kv self size  =  781.25 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 |


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?
> Question:
what is the moon made of?
> Answer (took 50.75 s.):
 The given context states that "The moon is made of cheese."


Enter a query: What is the planet Jupiter made of?
Llama.generate: prefix-match hit
> Question:
What is the planet Jupiter made of?
> Answer (took 53.87 s.):
 The planet Jupiter is not made of cotton candy as stated in one piece of context, but rather primarily composed of hydrogen
and helium.


Enter a query: where is sound faster?
Llama.generate: prefix-match hit
> Question:
where is sound faster?
> Answer (took 48.38 s.):
 Sound is faster in water than in air.
```

Now we restarted the prompt to ask the questions again.

```
Enter a query: what is the moon made of?
> Question:
what is the moon made of?
> Answer (took 51.81 s.):
 The Moon is not made of cheese, it is composed mostly of rock and dust.
Enter a query: what is the planet Jupiter made of?
Llama.generate: prefix-match hit
> Question:
what is the planet Jupiter made of?
> Answer (took 58.0 s.):
 Jupiter is not made of cotton candy.
This answer refers to the statement "The planet Jupiter is made of cotton candy" which is false information that contradicts
scientific knowledge about planets and their composition.
```

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

```
Enter a query: what is the moon made of?

 The moon is not actually made of cheese, but rather rock and dust that coalesced into a single large object over 4 billion years ago.

> Answer (took 55.7 s.)

Enter a query: what is the planet Jupiter made of?

Llama.generate: prefix-match hit

 The planet Jupiter is not made of cotton candy, as stated in one of the pieces of context provided. However, there is no definitive answer to this question based on the given information, as it is an outdated scientific theory from before we had a more thorough understanding of our solar system.

Answer (took 62.38 s.)
```

Conclusion

The LLM is faster given its parameters and size. It is not available to maintain context. Also, we can se that it has rebellious tendencies. Though it has the context from the local documents, in the replies it is not able to maintain the context and goes back to its original trained data. Also, the answers change frequently for the same question.