Model Name -tiiuae/falcon-7b-instruct

Model Type – falcon

License – Apache 2.0

Model Size – 14 GB

Reason – Falcon-7B is a strong base model, outperforming comparable open-source models. It has a good position on OpenLLM Leaderboard.

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

Info –Falcon-7B-Instruct is a 7B parameters causal decoder-only model built by TII based on Falcon-7B and finetuned on a mixture of chat/instruct datasets. This is trained on 1,500B tokens of RefinedWeb enhanced with curated corpora.

Local Dataset – falsefacts.txt **,** dataset_pointwise.pdf

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 for privateGPT we have additional steps to follow as the model available here is not available in out of the box supported format for privateGPT.

These are the models tried around falcon to make it work with privateGPT.

1.) Model - nomic-ai/gpt4all-falcon
2.) Updated .env to load this model. PrivateGPT was not able to support it. Some internal structure issues.

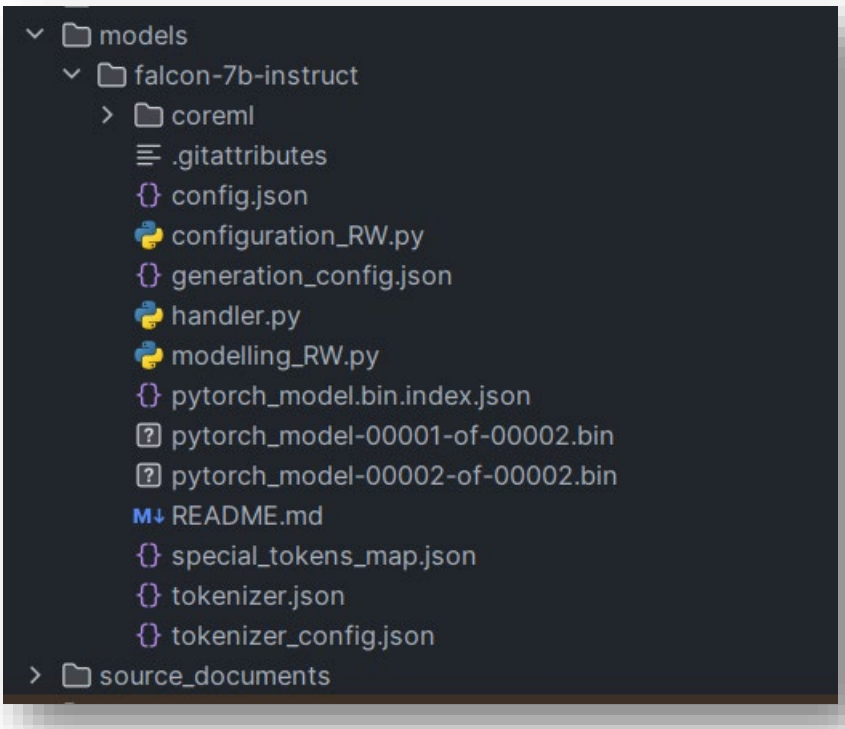
1.) Model - RachidAR/falcon-7B-ggml
2.) Updated .env to load this model. PrivateGPT was not able to support it. Some internal structure issues.

Conclusion – Failed due to python library incompatibility. Further study needed.

Here are the additional steps to make it work:-

1.) We need to install additional packages:-
    a. pip install eniops
    b. pip install xformers
2.) In privateGPT
    a. cd models/
3.) Clone the model into models/ directory.
    a. git lfs install
    b. git clone https://huggingface.co/tiiuae/falcon-7b-instruct
4.) In privateGPT.py make the following changes to support falcon:-
    a. In main method add an extra case for model type add these lines:-
        i. *llm = HuggingFacePipeline.from_model_id(model_id=model_path, task="textgeneration",model_kwargs={"temperature": 0, "max_length": 1000, "trust_remote_code": True})*
    b. At the top of privateGPT.py:
        i. *from langchain import HuggingFacePipeline*
5.) In the .env file set model name with absolute path of the folder containing model and set model_type = falcon

This is how privateGPT/models/ is populated now:-



Here’s how privateGPT.py looks now –

```
5   from langchain import HuggingFacePipeline
```

```
    case "GPT4All":
        llm = GPT4All(model=model_path, n_ctx=model_n_ctx, backend='gptj', n_batch=model_n_batch, callbacks=callbacks, verbose=False)
    case "falcon":
        llm = HuggingFacePipeline.from_model_id(model_id=model_path, task="text-generation",
                                                model_kwargs={"temperature": 0, "max_length": 1000,
                                                              "trust_remote_code": True})
    case _default:
        # raise exception if model_type is not supported
        raise Exception(f"Model type {model_type} is not supported. Please choose one of the following: LlamaCpp, GPT4All")
```

Here’s how .env file looks:

```
PERSIST_DIRECTORY=db
MODEL_TYPE=falcon
MODEL_PATH=E:/PycharmProjects/privateGPTFalcon7B/models/falcon-7b-instruct
EMBEDDINGS_MODEL_NAME=all-MiniLM-L6-v2
MODEL_N_CTX=1000
```

After this we proceed with training of the documents.

Training :-

```
(venv) PS E:\PycharmProjects\privateGPTFalcon7B> python .\ingest.py
Creating new vectorstore
Loading documents from source_documents
Loading new documents: 100%|████████████████████████| 2/2 [00:01<00:00,  1.02it/s]
Loaded 2 new documents from source_documents
Split into 18 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.

```
(venv) PS E:\PycharmProjects\privateGPTFalcon7B> python .\privateGPT.py -M
Using embedded DuckDB with persistence: data will be stored in: db
Loading checkpoint shards: 100%|████████████████████████████████| 2/2 [01:03<00:00, 31.58s/it]
Enter a query:
```

Inference:-

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

> Question:

what is the moon made of?

> Answer (took 89.03 s.):

The moon is made of cheese.

> Question:

how to make a sandwich?

> Answer (took 419.07 s.):

Go to the top of the page and click on the "Answer" button. Then, type in your answer and hit "Enter" to submit it.

Conclusion

The LLM did not yield very accurate results on local documents. Also, the resource utilisation for inference is very high. It consumes all of 32GB of memory and takes around 2 minutes average to give response.