How to create a private ChatGPT that interacts with your local documents

By **Ben Dickson** - June 1, 2023

As much as ChatGPT is convenient, it has its tradeoffs. The fact that it requires you to send your data over the internet can be a concern when it comes to privacy, especially if you're using confidential documents. Additionally, it requires a constant internet connection, which can be an issue in areas with poor connectivity.

Fortunately, there is an alternative. You can run your own local large language model (LLM), which puts you in control of your data and privacy. In this article, we will explore how to create a private ChatGPT that interacts with your local documents, giving you a powerful tool for answering questions and generating text without having to rely on OpenAI's servers. We will also look at PrivateGPT, a project that simplifies the process of creating a private LLM.

The private LLM structure

Let's start with a zoomed-out view of the components you need to create a local language model that can interact documents. Your local LLM will have a similar structure, but everything will be stored and run on your own computer:

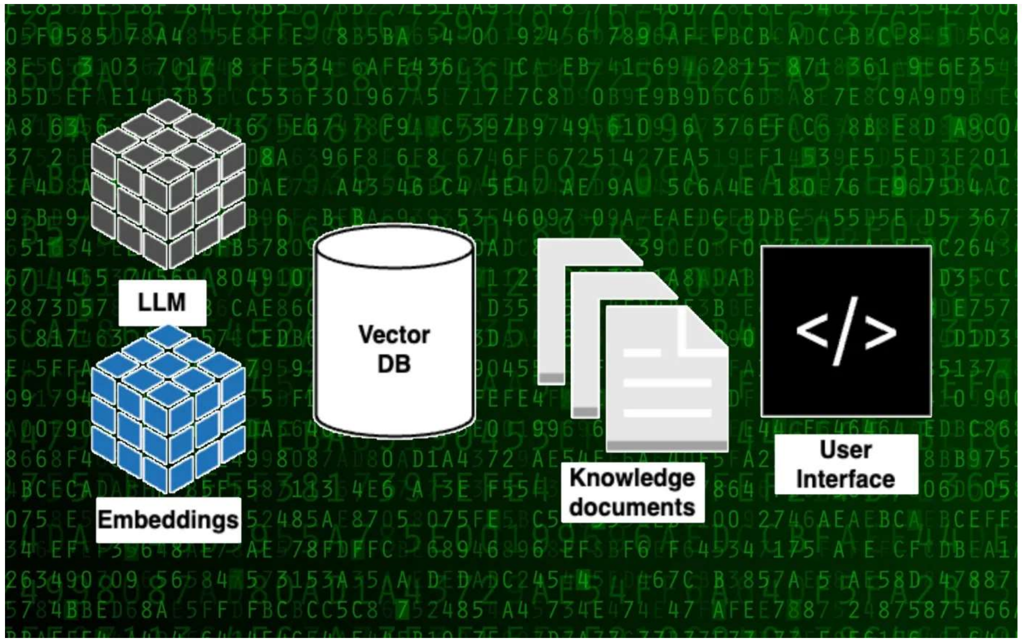
1. **Open-source LLM**: These are small open-source alternatives to ChatGPT that can be run on your local machine. Some popular examples include Dolly, Vicuna, GPT4All, and llama.cpp. These models are trained on large amounts of text and can generate high-quality responses to user prompts.

2. **Embedding model**: An embedding model is used to transform text data into a numerical format that can be easily compared to other text data. This is typically done using a technique called word or sentence embeddings, which represent text as dense vectors in a high-dimensional space. These embeddings can be used to find documents that are related to the user's prompt. The SentenceTransformers library contains a rich variety of pre-trained embedding models.

3. **Vector database**: A vector database is designed to store and retrieve embeddings. It can store the content of your documents in a format that can be easily compared to the user's prompt. Faiss is a library that you can use to add vector similarity comparisons on top of other data stores. But there are also a few open-source vector databases that you can install on your computer including Qdrant, Weaviate, and Milvus.

4. **Knowledge documents**: A collection of documents that contain the knowledge your LLM will use to answer your questions. These documents depend on your application. For example, it can be a collection of PDF or text documents that contain your personal blog posts.

5. **User interface**: The user interface layer will take user prompts and display the model's output. This can be a simple command-line interface (CLI) or a more sophisticated web application such as Streamlit. The user interface will send the user's prompt to the application and return he model's response to the user.



private LLM architecture

Private LLM workflow

Before you can use your local LLM, you must make a few preparations:

1. Create a list of documents that you want to use as your knowledge base

2. Break large documents into smaller chunks (around 500 words)

3. Create an embedding for each document chunk

4. Create a vector database that stores all the embeddings of the documents

If you add documents to your knowledge database in the future, you will have to update your vector database.

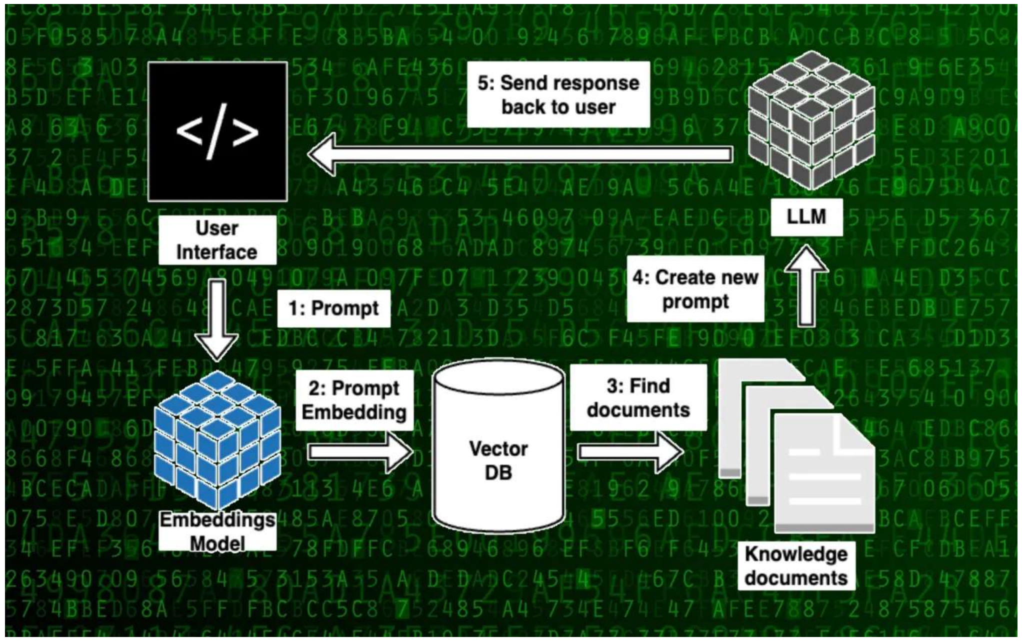
1. The user enters a prompt in the user interface.

2. The application uses the embedding model to create an embedding from the user's prompt and send it to the vector database.

3. The vector database returns a list of documents that are relevant to the prompt based on the similarity of their embeddings to the user's prompt.

4. The application creates a new prompt with the user's initial prompt and the retrieved documents as context and sends it to the local LLM.

5. The LLM produces the result along with citations from the context documents. The result is displayed in the user interface along with the sources.



private LLM workflow

Open-source LLMs are much smaller than state-of-the-art models like ChatGPT and Bard and might not match them in every possible task. But augmenting these language models with your own documents makes them very

PrivateGPT

By using a local language model and vector database, you can maintain control over your data and ensure privacy while still having access to powerful language processing capabilities. The process may require some technical expertise, but there are many resources available online to help you get started.

One solution is PrivateGPT, a project hosted on GitHub that brings together all the components mentioned above in an easy-to-install package. PrivateGPT includes a language model, an embedding model, a database for document embeddings, and a command-line interface. It supports several types of documents including plain text (.txt), comma-separated values (.csv), Word (.docx and .doc), PDF, Markdown (.md), HTML, Epub, and email files (.eml and .msg).

To use PrivateGPT, you'll need Python installed on your computer. You can start by cloning the PrivateGPT repository on your computer and install the requirements:

```
git clone https://github.com/imartinez/privateGPT.git
cd privateGPT/
pip install -r requirements.txt
```

Next, you need to download a pre-trained language model on your computer. Create a "models" folder in the PrivateGPT directory and move the model file to this folder. PrivateGPT is configured by default to work with GPT4ALL-J (you can download it here) but it also supports llama.cpp. These are both open-source LLMs that have been trained for instruction-following (like ChatGPT). They have also been designed to run on computers with consumer-grade hardware. Llama.cpp works especially well on Mac computers with M1 processors.

Next, you have to create your knowledge base. PrivateGPT has a "source_documents" folder where you must copy all your documents. After that, you must populate your vector database with the embedding values of your documents. Fortunately, the project has a script that performs the entire process of breaking documents into chunks, creating embeddings, and storing them in the vector database:

```
python ingest.py
```

project’s “db” folder. One thing to note is that LangChain needs to be connected to the internet to download the pre-trained embedding model. After that, all processing takes place on your own computer and you don’t need internet connectivity.

Depending on the number of documents that you have, creating the vector database might take several minutes. Once the preparation is finished, you can start the model with the following command:

```
python privateGPT.py
```

And then you can start talking to your local LLM with no strings attached. It will answer your questions and provide up to four sources from your knowledge base for each reply. PrivateGPT is an experimental project. It is not fast (it can take 20-30 seconds to respond) and is not optimized for every type of hardware. Its installation might also run into bugs based on your operating system and hardware. But it is surely a preview of one of the many directions the field is taking and the powerful applications that open-source LLMs can unlock.

Ben Dickson

Ben is a software engineer and the founder of TechTalks. He writes about technology, business and politics.