Documentación Proyecto Promtior

Martin Civetta

First, because I didn't have access to the OpenAI API, I decided to do the project with Ollama. I started by downloading Ollama locally to test it. Once I understood how it worked and after reading the documentation, I was able to link it to the RAG. Then I used the WebLoader framework to load the Promtior sites that would assist with the query. I loaded the main page and the /service site. I didn't use the soupstrainer framework because when I looked into the html in the websites, I couldn't differentiate the classes or elements that would help me from the ones that wouldn't as I couldn't identify a set structure. I also tried scraping an article, but it made the responses worse. I also loaded the pdf using pdfloader framework. I also had to use the hugging face embeddings because i did not have access to OpenAI's embeddings

Since I knew that having ollama running locally wouldn't work for a chatbot, I decided to dockerize an ollama image and pull the llama3.2 model. I knew that this didn't solve the issue but it helped me understand how to use ollama in a container. However this caused an issue later on because I was linking it wrong and I didnt know that llm could connect to despite being in a container.

Once I verified the responses were satisfactory, I converted the python app into an API so I could interact with it with a frontend. I decided to use the angular framework to design the frontend. I used the httpclient import to communicate with the API.

Once I was satisfied with the results I had to deploy the app. Even though it was not the best option, I used azure containers to host an image of ollama so I could connect outside of a localhost. I run the command "ollama pull llama3.2" inside the container. I say it wasnt the best option because it would be better to host an image of ollama with llama 3.2 already downloaded and the free tier of azure is pretty limited so it takes a long time to respond and it may run out of memory space after a few inquiries.

Then I had to deploy the api. I once again chose to use an azure container to host the api. I created an image of my application using the docker compose and tried to create an azure container. This time however, I received an error saying index.docker.io was down. What I decided to do was to create a container registry and push the image of my app there, and then create a container using the image hosted in the azure container registry.

After verifying that both containers worked, I hosted the frontend. First I tried using azure web app service, but realised it would not work. Instead I resolved to use azure's static web app service to host the frontend.
It's important to note that it is necessary to change the following setting in the static site in order to facilitate api communication. In this case, it's in google chrome

Domains:
Frontend: https://gentle-sea-04faef20f.5.azurestaticapps.net/
API container: 20.253.25.149:8000
Ollama container: 51.8.45.150:11434