A.I Project

Name⇒Faizan ul Haq
RollNo⇒I211771
Section⇒DS-M

Premise:

I attempted Q1 where we were assigned to create a chatbot that was to be expert on a single field or domain.
My Domain⇒Food Recipes
My Approach⇒ RAG (Retrieval Augmented Generation)

I took help of Langchain, Chroma db, Ollama to train the model on retrieval of specific inputs from the vector database
While Flask was used to deploy the project to a local server.

The Dataset:

The dataset i used was downloaded from kaggle the recipeNLG dataset.
It was a 2Gb csv consisting of more than a hundred thousand recipes.
Since the embeddings were taking too long, I took a subset of 3000 recipes to create the vector database.

Cleaning:

The dataset consisted of 5 features including **title, ingredients, directions, links, sources**
I dropped the links and sources
The rest of the columns were also cleaned for any unknown ascii texts, replaces the ascii for degree symbol with ‘degrees’ and ‘c.’ with ‘cups’
To create the vector database the data was then concatenated together and embedded and stored into the database

Experimenting:

I tried to experiment with the embeddings, i created a subset of 300 embeddings in 3 different ways

1) Title:””,\n ingredients:”” ,\n directions:””
2) Title:””, ingredients:”” , directions:””
3) Everything concatenated together into a big string

Then i created its embeddings and experimented it with the same inputs and tried finding out its cosine similarity

```
from tqdm import tqdm
datasets = [dataset1, dataset2, dataset3]
phiEmbedder = OllamaEmbeddings(model="phi3",show_progress=True)
for i,dataset in tqdm(enumerate(datasets)):
    vector_db = Chroma.from_documents(
        documents=dataset[0:300],
        embedding=phiEmbedder,
        collection_name="recipe",
        persist_directory=f"phidbb{i+1}"
    )
```

```
OllamaEmbeddings: 100%|██████████| 300/300 [50:33<00:00, 10.11s/it]
OllamaEmbeddings: 100%|██████████| 300/300 [50:16<00:00, 10.06s/it]
OllamaEmbeddings: 100%|██████████| 300/300 [49:16<00:00,  9.85s/it]
3it [2:30:24, 3008.13s/it]
```

Creating embeddings

The results of the cosine similarity:

```
originalResult vs phidbb1Result1: 0.32201349793031875
originalResult vs phidbb1Result2: 0.40833713578635994
originalResult vs phidbb2Result1: 0.3581934413376319
originalResult vs phidbb2Result2: 0.3679498519182043
originalResult vs phidbb3Result1: 0.4633686276961576
originalResult vs phidbb3Result2: 0.38801159182646217
==========
phidbb1Result1 vs phidbb1Result2: 0.8252220334858131
phidbb1Result1 vs phidbb2Result1: 0.7900233448105534
phidbb1Result1 vs phidbb2Result2: 0.7532117443016427
phidbb1Result1 vs phidbb3Result1: 0.6995680682079951
phidbb1Result1 vs phidbb3Result2: 0.8289385818779827
==========
phidbb1Result2 vs phidbb2Result1: 0.7918070206376462
phidbb1Result2 vs phidbb2Result2: 0.7090607554337315
phidbb1Result2 vs phidbb3Result1: 0.7305773871655125
phidbb1Result2 vs phidbb3Result2: 0.7693995676855514
==========
phidbb2Result1 vs phidbb2Result2: 0.7241184674119164
phidbb2Result1 vs phidbb3Result1: 0.7299342357174551
phidbb2Result1 vs phidbb3Result2: 0.750744386453635
==========
phidbb2Result2 vs phidbb3Result1: 0.6460447892490335
phidbb2Result2 vs phidbb3Result2: 0.6980160113081622
==========
phidbb3Result1 vs phidbb3Result2: 0.6518447360353664
==========
==========
```

The best results can be seen with the 3rd type.
And so i used it to create embeddings for the 3000 subset and use it for the data RAG model

How To Run:

The frontend will ask for 3 types of inputs
/recipe <food name> - Get a recipe for a specific dish
/recommend <ingredient names> - Get recommendations for ingredients you have
/how - ask the bot how much something is needed

These are the only inputs that I have constrained the users upon.