return-to-local-llm-w-RAG

<📚>

- Nishad's repository https://github.com/nishad/llm-workshop-notebooks
- Notes from the workshop last year
- ollama/ollama
  - docs/
    - linux.md: Linux uninstall info here
    - windows.md: Windows stuff here

(revisit) 01_using-ollama-with-python.ipynb

- I `pip install`-ed ollama in my WSL `.venv`
  - (I was running the notebook on my home desktop last time)
  - See pypi.org > ollama
- Hit trouble attempting to import and then use `ollama`
  - `ConnectionError`
  - My hypothesis is that I need to download Ollama separately from the pip install
  - A ha, per PyPI: "Ollama should be installed and running"
  - So `$ curl -fsSL https://ollama.com/install.sh | sh`
  - See ollama install details
  - 💥 💥 *This seems like a big "gotcha"* -- the notebook doesn't seem to address the fact that the separate download/install is needed
- OK now the cell with `available_models = ollama.list()` runs with no errors, but I don't get any models back.
  - Perhaps I need to "pull" a model to make it available first??
  - Back to the terminal...
  - `ollama run llama3.2`
  - This ^^^ is a relatively small, 2GB model with 1b params
  - See get a first model
- OK, at cell 01.05 it looks like we need IPython so I confirmed that I have this in my `.venv`

02_introduction-to-rag-with-ollama.ipynb

- Chroma actually does seem to *only* need the pip install
- Jinja also via pip, see Installation
- What does the following mean, actually??

> ```
> RAG combines the power of large language models with a vector database to enable dynamic and context-aware response generation based on external knowledge.
> ```

- "Make sure the `nomic-embed-text` embedding model is installed and running."
  - OK, I ran `$ ollama pull nomic-embed-text`
  - When I access `ollama.list()`, though, it doesn't show in the list...
  - So, it's not a model in the same way as `llama3.2`?
  - I noticed that for llama3.2 the command was `run` and for `nomic-embed-text` it was `pull`...
- 📩 "Learn more about embeddings and their applications in IBM's Documentation."
  - Yeah... I don't really understand what the concept is
  - Interesting, from the article ^^^:

> ```
> One model, Word2Vec (word to vector), developed by Google in 2013, is a method to efficiently create word embeddings by using a two-layer neural network. It takes as input a word and spits out an n-dimensional coordinate (the embedding vector) so that when you plot these word vectors in a three-dimensional space, synonyms cluster.
> ```

ChromaDB*STUFF*🛠️

- The documentation seems pretty good
- Persistent Client
  - Makes sense to me that if you want to generate responses based on the same text more than once, you wouldn't want to create embeddings for the text each time, so, using a persistent client
- `chromadb.utils.embedding_functions`
  - "Chroma provides a convenient wrapper around Ollama' s embeddings API. You can use the OllamaEmbeddingFunction embedding function to generate embeddings for your documents with a model of your choice."
  - See docs.trychroma.com > Ollama
  - `chromadb.utils.embedding_functions.OllamaEmbeddingFunction` - see OllamaEmbeddingFunction below

Python*STUFF*

(that I'd like to come back to at some point - this code seems well-written to me)

- list comprehensions -- see Python syntax bits below
- `upsert` !?!? Creates a dict? Does something with a dict?!?
- Lots of syntax that I don't immediately get in:
  - *funcs* `get_embedding_for_document(document)`, `load_documents_with_line_ids_and_embeddings(file_path, collection)`
- I keep seeing an index `[0]` at the end of things where I don't expect it... (random)
  - See for example using `query()` results below
- IPython seems to have neat tricks
  - There was a markdown thing in notebook 01 that seemed simple enough
  - The use of `IPython.display` for HTML in 02.07 is not something I understand A.T.M.
- `zip`, and I think another list comprehension -- see Python syntax bits below
- I will now be adding `Python` to all code blocks in notebook markdown!
- Also looks like Nishad is using doc strings for all functions, good reminder to do that!

Python syntax bits

```Python
# list comprehension - see 02.05
with open(file_path, 'r') as file:
    lines = file.readlines()
documents = [line.strip() for line in lines if line.strip()]
# zip - see 02.07
data = [[id_num, round(distance, 3), doc] for id_num, distance, doc in
zip(ids, distances, documents)]
```

02.07 Querying the Database for Matching Concepts

- This is where interesting stuff starts happening
- It helped me to understand the notes in the notebook when I realized that there are 892 lines in made-with-cc.txt, and each of these is vectorized ("Successfully added 892 documents with embeddings to the collection.") -- I think the notebook details may use "documents" and "paragraphs" interchangeably but ***I think*** both mean a line in made-with-cc.txt.
- Here's the `query(...)` method in Chroma docs...

02.10 RAG example

- Wait, so... The RAG gets us passages that are related to the query text...
- But, then we just tell the model to use those, and that's it!?
- I guess I thought that the content that was retrieved using the vector database would be incorporated in some different way than just telling the model to "answer using these references"??

OllamaEmbeddingFunction

Confused by the `url` arg...

```
chromadb.utils.embedding_functions.OllamaEmbeddingFunction(
    url="http://localhost:11434/api/embeddings"
    model_name="nomic-embed-text:latest"
)
# OK I was confused by the URL arg but localhost makes *a little* more sense
```

using `query()` results

I guess I'm confused because I don't know the structure of the QueryResult object...
I'm having trouble finding these details in docs...

```
# Extract results data
ids = results['ids'][0]  # Extract the IDs of the matching documents.
documents = results['documents'][0]  # Extract the document content of the
matches.
distances = results['distances'][0]  # Extract the distances between the query
and matches.
```

---

ollama install details

```
$ curl -fsSL https://ollama.com/install.sh | sh
>>> Installing ollama to /usr/local
[sudo] password for brieswsl:
>>> Downloading Linux amd64 bundle
######################################################################## 100.0%
>>> Creating ollama user...
>>> Adding ollama user to render group...
```

```
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
Created symlink /etc/systemd/system/default.target.wants/ollama.service →
/etc/systemd/system/ollama.service.
>>> The Ollama API is now available at 127.0.0.1:11434.
>>> Install complete. Run "ollama" from the command line.
```

get a first model

```
$ ollama run llama3.2
pulling manifest
pulling dde5aa3fc5ff... 100%
 ▕▏  2.0
GB
pulling 966de95ca8a6... 100%
 ▕▏  1.4
KB
pulling fcc5a6bec9da... 100%
 ▕▏  7.7
KB
pulling a70ff7e570d9... 100%
 ▕▏  6.0
KB
pulling 56bb8bd477a5... 100%
 ▕▏    96
B
pulling 34bb5ab01051... 100%
 ▕▏   561
B
verifying sha256 digest
writing manifest
success
```