



updated import according to langchain-ai/langc...

Extra measures for persistence refelcting chan...

Implemented persistency in vectorstore

RAG Service

config.py_example

nginx_example.conf

ragservice_unittest.py

ragservice.py

RAG Service is a template service for a retrieval-augmented generator based on the examples of LangChain. See: Build a Retrieval Augmented Generation (RAG) App

Technologies

















Initial commit









Installation

- 1. Clone L1Blom/rag to your project directory
- 2. Move config.py_example to config.py and add your API Key
- 3. Choose an ID for your instance, like MyDocs

last month

2 weeks ago

5 hours ago

12 hours ago

- 4. Copy constants/constants.py to constants MyDocs.py
- 5. Change the contents of this file to reflect your situation
- 6. Create a MyDocs/ directory in data/ and in MyDocs/ a directory vectorstore/
- 7. Add the files for your RAG context, like text-files, to your MyDocs/ directory
- 8. Create a Python virtual environment for your project (optional)
- 9. Modify the example services/rag.service_template file to point to the right place of your project directory
- 10. Copy the services file to /etc/systed/system/rag MyDocs.service
- 11. Enable and start the service

```
sudo systemctl enable rag_MyDocs
sudo systemctl start rag_MyDocs
sudo systemctl status rag_MyDocs
```

0

Commands

All calls support POST and GET. For <ID> use your chosen ID like MyDocs

1. /prompt/<ID>/

Parameter: prompt (string)

Your prompt to be send to openAl

2. /prompt/<ID>/model

Parameter: model (string)

Your model to be used, like "gpt-4o"

Checking on valid models with OpenAI client.models.list(). Can result in http 500 error (non-fatal)

3. /prompt/<ID>/temp

Parameter: temp (string, will be cast to float)

Temperature setting, between 0.0 and 2.0

Settings above 1.0 can give significant halicunations and degrades performance too.

Timeout can result in http 408 error (non-fatal)

4. /prompt/<ID>/reload

Parameters: none

After adding files to your data directory, use this to reload the vector store

5. /prompt/<ID>/clear

Paramters: none

Clears the cache, the in-memory history

6. /prompt/<ID>/cache

Paramaters: none

Prints the cache contents to the response object

7. /prompt/<ID>/image

Parameters: prompt (string), image (URL to image)

Uploads the image to openAI and use prompt to get the desired contents like: 'What is the mood of the persons?'

Note: only works if model is set to 'gpt-4o'. Other models result in http 500 error (non-fatal)

Usage

change the model
curl -X POST --data-urlencode "model=gpt-40" http://<your server>:<your port>/prompt/<ID>/model

```
Model set to: gpt-40
# prompt to your data
curl -X POST --data-urlencode "prompt=your question?" http://<your server>:<your port>/prompt/<ID>
Your answer based on the context files provided in data/<ID>
```

Constants file

14/08/2024, 21:12

```
Q
# simple string like "mydata"
ID="<your identifier>"
# any unused port, will run the Flask server
PORT=9100
# Set to True if you want to enable Flask debug
DEBUG=False
# Any other level will make it less verbose
LOGGING_LEVEL="INFO"
# Where the files reside for unit testing
HTML="data/_unittest/html"
# Directory that will be scanned for files to be added to the context
DATA_DIR="/home/data"
# All the file extentions you want to be part of the context, see LangChain documentation
DATA_GLOB="*.txt"
# The lower, the more precise. Needs to be between 0.0 and 2.0
TEMPERATURE=0.0
# Influences the way answers are produced
contextualize_q_system_prompt = (
        "Given a chat history and the latest user question "
        "which might reference context in the chat history, "
        "formulate a standalone question which can be understood "
# Influences the way answers are produced
system_prompt = ( # choose your language, English works best
        "You are a chatbot and gives answer in not more than 3 sentences"
chunk size=1000
                  # depending on your data, seel LangChain documentation
chunk_overlap=100 # idem
```

Unit tests

To run the unit tests, run the program in the project directory using the ID '_unittest'. It will start a local RAG service accessible at port 8888 (see constants__unittest.py for all defaults).

```
<your virtual environment>/bin/python ragservice.py _unittest
INFO:httpx:HTTP Request: GET https://api.openai.com/v1/models "HTTP/1.1 200 OK"
INFO:root:Context loaded from 1 documents
INFO:chromadb.telemetry.product.posthog:Anonymized telemetry enabled. See https://docs.trychroma.
INFO:httpx:HTTP Request: POST https://api.openai.com/v1/embeddings "HTTP/1.1 200 OK"
INFO:root:Chain initialized: gpt-40
  * Serving Flask app 'ragservice'
  * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
  * Running on all addresses (0.0.0.0)
  * Running on http://127.0.0.1:8888
  * Running on http://192.168.2.200:8888
INFO:werkzeug:Press CTRL+C to quit
```

Now you are able to run the unit tests:

```
<your virtual environament>/bin/python ragservice_unittest.py -v
test_chache_content (__main__.RagServiceMethods.test_chache_content)
Test to print the contents of the cache ... ok
test_clear (__main__.RagServiceMethods.test_clear)
Test to clear the cache ... ok
test_image (__main__.RagServiceMethods.test_image)
Test image ... ok
test_model (__main__.RagServiceMethods.test_model)
Test model setting, correct or incorrect model according to OpenAI ... ok
test_prompt (__main__.RagServiceMethods.test_prompt)
Test prompt ... ok
test_reload (__main__.RagServiceMethods.test_reload)
Test reload of the data ... ok
```

```
test_temperature (__main__.RagServiceMethods.test_temperature)
Test to set temparature too high, low, within boundaries 0.0 and 2.0 ... ok

Ran 7 tests in 5.923s
```

TODO's and wishes

- More file formats like PDF
- · Upload files
- · Other API's than OpenAI and other LLM's
- A simple frontend from the GitHub community

Contributing

Pull requests are welcome. For major changes, please open an issue first to discuss what you would like to change.

Please make sure to update tests as appropriate.

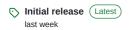
License

MIT

The image used in the unittest is licensed <u>CC BY-NC-ND 4.0</u> and was found at <u>Trusted Reviews</u>



Releases 1



Packages

No packages published
Publish your first package

Languages

Python 94.0%
 Shell 6.0%