

AI Project

Configuration:

```
1  services:
2    mongo:
3      image: mongo:latest
4      container_name: "llm_engineering_mongo"
5      logging:
6      options:
7        max-size: 1g
8    environment:
9      MONGO_INITDB_ROOT_USERNAME: "llm_engineering"
10     MONGO_INITDB_ROOT_PASSWORD: "llm_engineering"
11    ports:
12      - 27017:27017
13    volumes:
14      - mongo_data:/data/db
15    networks:
16      - local
17    restart: always
18
19    qdrant:
20      image: qdrant/qdrant:latest
21      container_name: "llm_engineering_qdrant"
22      ports:
23        - 6333:6333
```

```
rag-py3.12(base) komalbagwe@10-17-99-138 RAG % /Users/komalbagwe/Library/Caches/pypoetry/virtualenvs/rag-KP5L7G78-py3.12/bin/python /Users/komalbagwe/Documents/RAG/run_tasks.py
Running command: docker-compose up -d
Container llm_engineering_mongo Running
Container llm_engineering_qdrant Running

Running command: clearml-init
ClearML SDK setup process
Configuration file already exists: /Users/komalbagwe/clearml.conf
Leaving setup, feel free to edit the configuration file.

Running command: python gradio_interface.py
```

Ln 21, Col 1 Spaces: 4 UTF-8 LF Python 3.12.2 ('rag-KP5L7G78-py3.12': F

ETL Pipeline:

```

from clearml import Task
from pymongo import MongoClient
from bs4 import BeautifulSoup
import requests
from youtube_transcript_api import YouTubeTranscriptApi
import logging

# MongoDB connection
mongo_client = MongoClient('mongodb://llm_engineering:llm_engineering@127.0.0.1:27017')
db = mongo_client['rag_system']
collection = db['raw_data']

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

def etl_pipeline():
    # Initialize the task with the correct project name
    task = Task.init(project_name="RAG_System", task_name="ETL Task")
    task_logger = task.get_logger()

    def extract_ros2_docs():
        ros2_docs_urls = [
            "https://www.ros.org/",
            "https://github.com/ros-navigation/navigation2",
            "https://github.com/bmaxdk/R0S2-Nav2-with-SLAM-and-Navigation",
            "https://github.com/moveit/moveit2",
            "https://github.com/ros-simulation/gazebo_ros_pkgs",
            "https://docs.nav2.org/",
            "https://moveit.ai/",
            "https://gazebo.sim.org/home",
            "https://github.com/Medissaoui07/Autonomus-Navigation-Robot-R0S2",
            "https://github.com/ros-navigation/navigation2",
            "https://github.com/moveit/moveit2",
            "https://github.com/Andrej0rsula/ign_moveit2_examples/blob/master/docs/README.md",
            "https://github.com/oKermorgant/gz_moveit2_examples",
            "https://medium.com/schmiedeone/getting-started-with-ros2-part-1-d4c3b7335c71",
            "https://medium.com/@tetraengnrrng/a-beginners-guide-to-ros2-29721dcf49c8",
            "https://medium.com/@tetraengnrrng/a-beginners-guide-to-ros2-29721dcf49c8",
            "https://medium.com/robotics-zone/understanding-moveit2-and-ros-2-for-robotics-61d74832cd2a",
            "https://medium.com/ros2-basics/building-your-own-ros2-navigation-stack-from-scratch-cc9c6b6a32e6",
            "https://medium.com/robotics-zone/introducing-ros2-and-gazebo-simulation-for-robotics-3e4054d4f8f8",
            "https://medium.com/robotics-zone/using-ros2-and-gazebo-to-simulate-robots-in-a-vibrant-world-34a6a4f35b28"
        ]

        for url in ros2_docs_urls:
            try:
                response = requests.get(url)
                soup = BeautifulSoup(response.content, 'html.parser')
                text = soup.get_text()
                data = {'source': 'github', 'url': url, 'content': text}
                collection.insert_one(data)
                task_logger.report_text(f"Extracted and stored data from {url}")
                print(f"Extracted and stored data from {url}") # Use standard logger
            except Exception as e:
                task_logger.report_text(f"Failed to extract data from {url}: {e}")
                print(f"Failed to extract data from {url}: {e}") # Use standard logger

    def extract_youtube_videos():
        youtube_video_ids = ['7TVWtLADxwRw', 'rtrGoGsMVLi&list=PLGg0XDQjCkksJDPhXsFU_RIqEh08nG0V', 'sVUKehMBtpQ', "Xbij9Tst-WA", "jkoGkAd0GYk", "QI"]

        for video_id in youtube_video_ids:
            try:
                transcript = YouTubeTranscriptApi.get_transcript(video_id)
                transcript_text = ' '.join([t['text'] for t in transcript])
                data = {'source': 'youtube', 'url': f"https://www.youtube.com/watch?v={video_id}", 'content': transcript_text}
                collection.insert_one(data)
                task_logger.report_text(f"Extracted and stored data from YouTube video {video_id}")
                print(f"Extracted and stored data from YouTube video {video_id}")
            except Exception as e:
                task_logger.report_text(f"Failed to extract data from YouTube video {video_id}: {e}")
                print(f"Failed to extract data from YouTube video {video_id}: {e}")

```

```
def extract_ros2_docs():
    "https://medium.com/robotics-zone/understanding-moveit2-and-ros-2-for-robotics-61d74832cd2a",
    "https://medium.com/ros2-basics/building-your-own-ros2-navigation-stack-from-scratch-cc9c6b6a32e6",
    "https://medium.com/robotics-zone/introducing-ros2-and-gazebo-simulation-for-robotics-3e4054d4f8f8",
    "https://medium.com/robotics-zone/using-ros2-and-gazebo-to-simulate-robots-in-a-vibrant-world-34a6a4f35b28"

    for url in ros2_docs_urls:
        try:
            response = requests.get(url)
            soup = BeautifulSoup(response.content, 'html.parser')
            text = soup.get_text()
            data = {'source': 'github', 'url': url, 'content': text}
            collection.insert_one(data)
            task_logger.report_text(f"Extracted and stored data from {url}")
            print(f"Extracted and stored data from {url}") # Use standard logger
        except Exception as e:
            task_logger.report_text(f"Failed to extract data from {url}: {e}")
            print(f"Failed to extract data from {url}: {e}") # Use standard logger

def extract_youtube_videos():
    youtube_video_ids = ['7TVwLADxwRw', 'rtrGoGsMvLI&list=PLgG0XDQqJckkSJDPhXsFU_RIqEh08nG0V', "sVUKeHMBtpQ", "Xbij9Tst-WA", "jkoGkAd0GYk", "QI

    for video_id in youtube_video_ids:
        try:
            transcript = YouTubeTranscriptApi.get_transcript(video_id)
            transcript_text = ' '.join([t['text'] for t in transcript])
            data = {'source': 'youtube', 'url': f"https://www.youtube.com/watch?v={video_id}", 'content': transcript_text}
            collection.insert_one(data)
            task_logger.report_text(f"Extracted and stored data from YouTube video {video_id}")
            print(f"Extracted and stored data from YouTube video {video_id}")
        except Exception as e:
            task_logger.report_text(f"Failed to extract data from YouTube video {video_id}: {e}")
            print(f"Failed to extract data from YouTube video {video_id}: {e}")
```

```
def etl_pipeline():
    def extract_youtube_videos():
        print(f"Failed to extract data from YouTube video {video_id}: {e}")

    def print_ingested_urls():
        urls = [doc['url'] for doc in collection.find()]
        print("Ingested URLs:")
        print("Ingested URLs:")
        for url in urls:
            logger.info(url)
            print(url)

    # Execute ETL steps
    extract_ros2_docs()
    extract_youtube_videos()
    print_ingested_urls()

    task.close()

# Directly execute the ETL pipeline locally
if __name__ == "__main__":
    etl_pipeline()
```

EXPERIMENTS LIST

SORT BY

ETL Task

Completed

Updated a minute ago

Created by Komal Bagwe

ETL Task

Draft

Updated 2 days ago

Created by Komal Bagwe

ETL Task

Draft

Updated 2 days ago

Created by Komal Bagwe

ETL Task

Draft

Updated 2 days ago

Created by Komal Bagwe

ETL Task

Draft

Updated 2 days ago

Created by Komal Bagwe

ETL Task

Failed

Updated 2 days ago

Created by Komal Bagwe

ETL Task

+ ADD TAB

EXECUTION

CONFIGURATION

ARTIFACTS

INFO

CONSOLE

SCALARS

PLOTS

DEBUG SAMPLES

Hostname: Komals-Air.Jan

Download full log

Filter: Filter By Regex

2024-12-08 19:39:20

ClearML Task: overwriting (reusing) task id=91fd2fb6a8294594aec5fa79decac054
2024-12-08 19:39:20,968 - clearml.Task - INFO - No repository found, storing script code instead
ClearML results page: <https://app.clear.ml/projects/6ec44870862748ee9e31b5651af0d929/experiments/91fd2fb6a8294594aec5fa79decac054>
Extracted and stored data from <https://www.ros.org/>

2024-12-08 19:39:21

Extracted and stored data from <https://www.ros.org/>
Extracted and stored data from <https://github.com/ros-navigation/navigation2>
Extracted and stored data from <https://github.com/ros-navigation/navigation2>

2024-12-08 19:39:23

ClearML Monitor: GPU monitoring failed getting GPU reading, switching off GPU monitoring
Extracted and stored data from <https://github.com/bmaxdk/R052-Nav2-with-SLAM-and-Navigation>
Extracted and stored data from <https://github.com/bmaxdk/R052-Nav2-with-SLAM-and-Navigation>

2024-12-08 19:39:24

Extracted and stored data from <https://github.com/moveit/moveit2>
Extracted and stored data from <https://github.com/moveit/moveit2>
Extracted and stored data from https://github.com/ros-simulation/gazebo_ros_pkgs
Extracted and stored data from https://github.com/ros-simulation/gazebo_ros_pkgs
Extracted and stored data from <https://docs.nav2.org/>
Extracted and stored data from <https://docs.nav2.org/>

2024-12-08 19:39:25

Extracted and stored data from <https://moveit.ai/>
Extracted and stored data from <https://moveit.ai/>
Extracted and stored data from <https://gazebosim.org/home>
Extracted and stored data from <https://gazebosim.org/home>
Extracted and stored data from <https://github.com/Medissaoui07/Autonomous-Navigation-Robot-R052>
Extracted and stored data from <https://github.com/Medissaoui07/Autonomous-Navigation-Robot-R052>

Jump to end

Feature Engineering:

fetch_similar_data

No results

```
from transformers import BertTokenizer, BertModel
import torch
import qdrant_client
from qdrant_client import models
from pymongo import MongoClient
import uuid
from bson import ObjectId
from qdrant_client.models import VectorParams, Distance

mongo_client = MongoClient('mongodb://llm_engineering:llm_engineering@127.0.0.1:27017')
db = mongo_client['rag_system']
collection = db['raw_data']

qdrant = qdrant_client.QdrantClient("http://127.0.0.1:6333")

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

collection_name = "raw_data"
collections_response = qdrant.get_collections()

if not any(col.name == collection_name for col in collections_response.collections):
    qdrant.create_collection(
        collection_name=collection_name,
        vectors_config=VectorParams(
            size=768,
            distance=Distance.COSINE
        )
    )

def extract_features(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True)
    with torch.no_grad():
        outputs = model(**inputs)
    embeddings = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
    return embeddings
```

```
vectors_config=VectorParams(
    size=768,
    distance=Distance.COSINE
)

def extract_features(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True)
    with torch.no_grad():
        outputs = model(**inputs)
    embeddings = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
    return embeddings

def push_features_to_qdrant():
    for doc in collection.find():
        content = doc['content']
        embedding = extract_features(content)

        point_id = str(uuid.uuid5(uuid.NAMESPACE_OID, str(doc['_id'])))

        payload = {
            "embedding": embedding.tolist(),
            "metadata": {"source": doc['source'], "url": doc['url']}
        }

        point = models.PointStruct(id=point_id, vector=embedding.tolist(), payload=payload)
        qdrant.upsert(collection_name=collection_name, points=[point])
        print(f"Features for {doc['url']} pushed to Qdrant.")

push_features_to_qdrant()
```

Point 00139b3b-4f42-5234-a461-5c0557eba77a

Payload:

embedding

~ [768 Items

- > [...] 100 Items
- > [...] 100 Items
- > [...] 100 Items
- > [...] 100 Items
- > [...] 100 Items
- > [...] 100 Items
- > [...] 100 Items
- > [...] 100 Items
- > [...] 68 Items

]

metadata

~ { 2 Items

```
"source": "github"
"url": "https://github.com/oKermorgant/gz_moveit2_examples"
}
```

Vectors:

FineTuning:

```
vectors_config=VectorParams(
    size=768,
    distance=Distance.COSINE
)

def extract_features(text):
    inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True)
    with torch.no_grad():
        outputs = model(**inputs)
    embeddings = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
    return embeddings

def push_features_to_qdrant():
    for doc in collection.find():
        content = doc['content']
        embedding = extract_features(content)

        point_id = str(uuid.uuid5(uuid.NAMESPACE_OID, str(doc['_id'])))

        payload = {
            "embedding": embedding.tolist(),
            "metadata": {"source": doc['source'], "url": doc['url']}
        }

        point = models.PointStruct(id=point_id, vector=embedding.tolist(), payload=payload)
        qdrant.upsert(collection_name=collection_name, points=[point])
        print(f"Features for {doc['url']} pushed to Qdrant.")

push_features_to_qdrant()
```

```
learning_rate=2e-5,
per_device_train_batch_size=1,
gradient_accumulation_steps=16,
num_train_epochs=3,
weight_decay=0.01,
save_strategy="epoch",
logging_dir="./logs",
logging_steps=10,
fp16=False,
report_to="clearml",
optim="adamw_torch",
max_grad_norm=0.3,
)

torch.cuda.empty_cache()

model.gradient_checkpointing_enable()

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
    data_collator=data_collator,
)

try:
    trainer.train()

    model.save_pretrained("./fine_tuned_llama2")
    tokenizer.save_pretrained("./fine_tuned_llama2")
    print("Fine-tuned model saved successfully.")
except RuntimeError as e:
    print(f"An error occurred during training: {e}")
    print("Consider further reducing batch size, sequence length, or using a smaller model.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

- Waiting for previous model to upload (2 pending, https://files.clear.ml/HuggingFace Transformers/Trainer.d6679e7829b744558
19e75b2b1ede8d8/models/checkpoint-3.zip)
2024-12-08 23:01:17,629 - clearml.Task - INFO
- Waiting for previous model to upload (2 pending, https://files.clear.ml/HuggingFace Transformers/Trainer.d6679e7829b744558
19e75b2b1ede8d8/models/checkpoint-3.zip)
100% | 1424.25/1424.25 MB [09:34<00:00, 2.48MB/s]:
2024-12-08 23:01:25,021 - clearml.Task - INFO - Completed model upload to https://files.clear.ml/HuggingFace%20Transformers/T
rainer.d6679e7829b74455819e75b2b1ede8d8/models/checkpoint-2.zip
2024-12-08 23:01:25,900 - clearml.storage - INFO - Uploading: 1424.25MB to /var/folders/66/q21rnv5143b58h0kjlw8rgmr0000gn/T/m
odel_package.fibb8806.zip
{'train_runtime': 1196.2704, 'train_samples_per_second': 0.008, 'train_steps_per_second': 0.003, 'train_loss': 0.425259470939
63623, 'epoch': 3.0}
100%| ClearML Monitor: Reporting detected, revertin
g back to iteration based reporting 135.00/1424.25 MB [00:49<07:55, 2.71MB/s]:
98% | 1395.00/1424.25 MB [09:28
98% | 1400.00/1424.25 MB [09:34
99% | 1405.00/1424.25 MB [09:39
99% | 1410.00/1424.25 MB [09:41
99% | 1415.00/1424.25 MB [09:44
100% | 1424.25/1424.25 MB [10:02<00:00, 2.36MB/s]:
100% | 1424.25/1424.25 MB [09:48<00:00, 1.91MB/s]:
```


Output:

RAG-based Query System

This system uses Qdrant to fetch context based on the query and generate a response using GPT-2 or another language model.

Enter your query

what is ros2?

 1

Clear

Submit

Generated Response

Question: what is ros2?

Answer: ros2 is an Android-based application and the Linux/iOS framework with Android-like developer interfaces. I used ros2 in Android 4.7 and Android v2.0, which may vary from what's listed when you start up the app.

This question is from the same developer as answer:

Question: how do you choose what to install?

Answer: ros2 is a minimal library. You may install it as a utility by adding one of the following files:

root-android -f install ros2 --output=root --build=android/androideabi-4/androideabi-4.5.0.tar.gz

For example, to setup two apps with different UI and UI elements, you could add one of the following:

root-android --debug=Debug

Flag