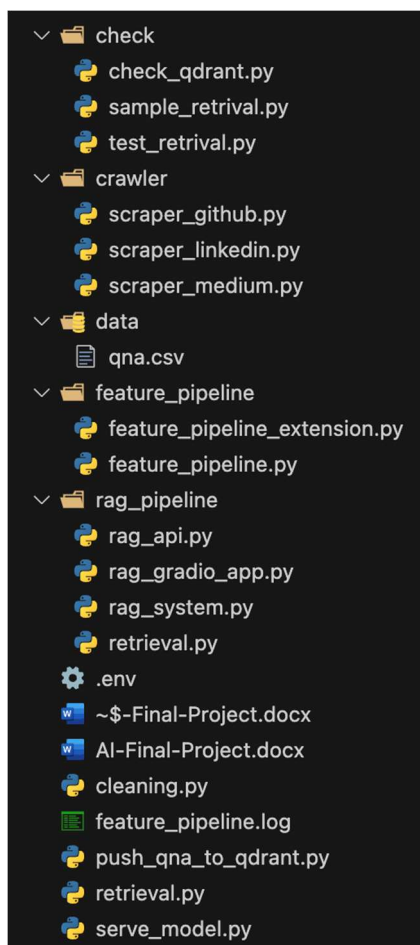


# CS-GY 6613 Final Project

- **Project Overview:**
    - Briefly introduce the concept of Retrieval Augmented Generation (RAG) systems.
    - Explain the core idea: combining retrieval-based and generation-based models to create a system that answers domain-specific questions using both large-scale information retrieval and language generation.
    - Outline the domain of your project: ROS2 robotics, navigation stack, and the various subdomains like ROS2 middleware, Nav2 navigation, motion planning, and simulation.
  - **Objective:**
    - The goal of this project is to build a proof-of-concept (PoC) RAG system that can answer detailed questions related to ROS2 robotics subdomains using a mix of document retrieval and model-based text generation.
    - Mention that the report will focus on the milestones, challenges, architecture, and implementation of the system.
- 

## 2. System Architecture

- **Structure:**



- **System Overview:**
    - Provide an overview of the architecture, explaining how different components interact to form the RAG system. Mention each major module:
      - **Crawler/Scraper:** Gathers raw data from GitHub repositories, Medium articles, and LinkedIn profiles related to ROS2 robotics.
      - **Cleaning:** Raw data is cleaned and stored in MongoDB.
      - **Featurization:** Converts the cleaned data into question-answer pairs stored in both MongoDB and Qdrant vector database.
      - **Check:** Checks Qdrant retrieval and connection.
      - **Serving Model:** The fine-tuned Hugging Face model is used to generate responses.
      - **Gradio App:** Allows users to interact with the system by selecting questions from a dropdown menu.
      - **ClearML:** Orchestrates and tracks experiments related to data ingestion and model training.
  - **Flow of Data:**
    - Describe the data flow in the system starting from data scraping, cleaning, storing in MongoDB, converting to question-answer pairs, storing in Qdrant, model fine-tuning, and final serving via the Gradio app.
  - **Components Diagram:**
    - Include a diagram of your system architecture (you can use the screenshot you attached earlier).
- 

### 3. Milestones & Deliverables

#### 1. Environment and Tooling Milestone

- **Objective:** Set up a Docker Compose environment for the development of the RAG system.
- **Deliverables:**
  - Docker Compose file with containers for MongoDB, Qdrant, ClearML, and the app (PyTorch/TF models).
  - Screenshot showing all services running.
- **Setup Details:**
  - Explain how you configured the Docker setup for the necessary components and dependencies.

#### 2. ETL Milestone

- **Objective:** Implement an ETL pipeline to ingest multiple media sources like GitHub repositories, YouTube videos, and Medium articles.
- **Details:**
  - Discuss the structure of your scrapers (e.g., `scraper_github`, `scraper_medium`, `scraper_linkedin`) and how they fetch data.
  - Describe the cleaning pipeline and how the raw data is stored in MongoDB.
  - Provide examples of scraped data and their storage format.

### 3. Featurization Pipeline Milestone

- **Objective:** Process raw data into question-answer pairs and store them in MongoDB and Qdrant.
- **Details:**
  - Describe how `feature_pipeline.py` and related scripts convert raw data into question-answer pairs.
  - Explain the usage of OpenAI API (if available) or fallback to manually created question-answer pairs stored in a `qna.csv` file.
  - Discuss how the featurized data is stored in MongoDB and pushed into the Qdrant vector database.

### 4. Fine-tuning Milestone

- **Objective:** Fine-tune a model for domain-specific question answering.
- **Details:**
  - Explain your approach to fine-tuning the Hugging Face model (based on GPT-2 or other base models).
  - If you didn't have access to OpenAI API, explain how you generated data for fine-tuning and used a self-made dataset.

### 5. Deploying the App Milestone

- **Objective:** Develop and deploy a Gradio app to allow users to interact with the RAG system.
  - **Details:**
    - Describe the Gradio app's functionality, including the dropdown interface for users to select questions.
    - Explain how the app connects to the Qdrant vector database to retrieve the top 5 most similar question-answer pairs.
    - Describe how the fine-tuned model is used to generate a coherent and accurate answer.
  - **Screenshots:** Provide screenshots of the Gradio app in action, showing the user interface and example interactions.
- 

## 4. Detailed Component Explanation

### 1. Crawler/Scraper Modules

- **Purpose:** Collect domain-specific data from external sources.
- **Components:** `scraper_github`, `scraper_medium`, `scraper_linkedin`.
- **Explanation:** Describe the functionality of each scraper module, the types of data collected, and how it's stored in MongoDB for further processing.

## 2. Data Cleaning Module (`cleaning.py`)

- **Purpose:** Clean and preprocess raw data before further usage.
- **Process:** Describe the data cleaning steps (e.g., removing irrelevant content, normalizing formats, handling missing data) and the final output that is stored in MongoDB.

## 3. Featurization Pipeline

- **Purpose:** Convert cleaned data into question-answer pairs suitable for the RAG model.
- **Process:** Describe how the featurization is done using OpenAI (if available) or manual data (from `qna.csv`), and how the question-answer pairs are stored in MongoDB and Qdrant.

## 4. Model Serving (`serve_model.py`)

- **Purpose:** Load and serve the fine-tuned model to answer user queries.
- **Details:** Explain how `serve_model.py` loads the fine-tuned Hugging Face model and uses it to generate answers based on the question retrieved from the Qdrant vector database.

## 5. Gradio App (`rag_pipeline/_gradio_app.py`)

- **Purpose:** Allow users to interact with the RAG system through a web interface.
  - **Details:** Describe how the Gradio app works, including how it takes input from users, queries the Qdrant vector database, and uses the fine-tuned model to generate answers.
-

## 5. Results and Evaluation

### Firstly, the output for the requested questions

RAG System Interface

Select a Question

Tell me how can I navigate to a specific pose - include replanning aspects in your answer.

Ask

Relevant Q&A Pairs

Q. What is the next step after completing the tutorial on movable joints? A: The next step is to add physical properties to the model or use xacro to simplify the URDF code.

Q. How do you move the camera in RViz? A: Use the Move Camera tool to change the viewpoint in the 3D view. This can be done by clicking and dragging inside the 3D window.

Q. How can you view the structure of an action type? A: Use the command: ros2 interface show

Q. How do you finish a visual robot model? A: "You finish the model by adding more shapes

Q. What is the type of joint used for the head in the robot model? A: The head is connected with a continuous joint, allowing it to rotate infinitely in either direction around the z-axis.

Final Answer

Tell me how can I navigate to a specific pose - include replanning aspects in your answer. For example, you can use the 'ros2 component list' command to list all available joints in a ROS 2 system. In this case, the controller is 'carrot1' and the turtle is moving relative to 'turtle2'. This allows you to see the movement of the joint in RViz. It also shows the relationship between the two joints, which is useful for visualizing joint states and their relationship to one another. The visual model is also updated to support more complex joints like the torso and leg joints. These updates make it easier to understand how joints behave in different environments, such as near-infrared and ultrasonic sensors.

RAG System Interface

Select a Question

Can you provide me with code for this task?

Ask

Relevant Q&A Pairs

Q. How can you find the value of a specific parameter? A: Use 'ros2 param get' to get its type and current value.

Q. How do you build and run the code in this tutorial? A: You build the code using 'colcon build' and then run the node with the command 'ros2 run cpp\_parameter\_event\_handler\_parameter\_event\_handler'.

Q. How do you create a custom service with three integers as input and one integer as output? A: In the 'srv' directory, create a file called 'AddThreeInts.srv' with the request 'int64 a;int64 b;int64 c' and response 'int64 sum'.











Q. How do you update the position of the shape based on the message? A: "In processMessage

Q. How do you run the example in this tutorial? A: "You can run the example by executing 'docker compose -f compose.deployment.yaml up' in the terminal

Final Answer

Can you provide me with code for this task? Answer: Yes, you can execute code like this in a ROS 2 launch file, but it is not possible to execute in C++. Instead, use 'ament\_target\_launch.py' or similar commands to get the parameters for the service and service nodes. This allows you to define custom actions for both services and their arguments. For example, for adding a timestamp parameter 'time\_to\_timestamp' will return 'touched\_time' once every 0.5 seconds." This is useful for specifying the duration to wait\_for\_all\_service and 'timer\_callback' parameters when creating a service or calling a callback function. Parameters can be modified dynamically during execution, such as adding 'rclcpp::parameters\_...

### Secondly, the docker containers for the Qdrant and MongoDB

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Disk read/write	Network I/O	PIDS	Actions
<input type="checkbox"/>	 mongodb	5862e070b3b2	mongo:<no>	27017:27017 	2.03%	157.6MB / 7.65GI	2.01%	233KB / 38.6MB	15.1MB / 17.9MB	43	  
<input type="checkbox"/>	 qdrant	fdf4cf23d886	qdrant/qdr:	6333:6333 	0.36%	399.3MB / 7.65GI	5.09%	193KB / 29.2MB	42.4MB / 41.7KB	52	  

### Thirdly, the python files hosting the model - serve\_model.py

```
(Desktop/AI-FINAL) $ python3 serve_model.py
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/bitsandbytes/cextension.py:34: UserWarning: The installed version of bitsandbytes was compiled without GPU support. 8-bit optimizers, 8-bit multiplication, and GPU quantization are unavailable.
  warn("The installed version of bitsandbytes was compiled without GPU support. ")
'NoneType' object has no attribute 'cadam32bit_grad_fp32'
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/peft/tuners/lora/layer.py:1264: UserWarning: fan_in_fan_out is set to False but the target module is 'Conv1D'. Setting fan_in_fan_out to True.
  warnings.warn(
* Serving Flask app 'serve_model'
* Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.1.234:5001
INFO:werkzeug:Press CTRL+C to quit
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
INFO:werkzeug:127.0.0.1 - - [08/Dec/2024 23:39:49] "POST /generate HTTP/1.1" 200 -
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
INFO:werkzeug:127.0.0.1 - - [08/Dec/2024 23:40:22] "POST /generate HTTP/1.1" 200 -
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
INFO:werkzeug:127.0.0.1 - - [08/Dec/2024 23:41:01] "POST /generate HTTP/1.1" 200 -
```

## rag\_api.py

```
(Desktop/AI-FINAL) $ python3 rag_pipeline/rag_api.py
INFO:sentence_transformers.SentenceTransformer:Use pytorch device_name: mps
INFO:sentence_transformers.SentenceTransformer:Load pretrained SentenceTransformer: sentence-transformers/all-MiniLM-L6-v2
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): huggingface.co:443
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/modules.json HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/config_sentence_transformers.json HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/README.md HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/modules.json HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/sentence_bert_config.json HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/adapters_config.json HTTP/11" 404 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/config.json HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /sentence-transformers/all-MiniLM-L6-v2/resolve/main/tokenizer_config.json HTTP/11" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "GET /api/models/sentence-transformers/all-MiniLM-L6-v2/revision/main HTTP/11" 200 6694
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "GET /api/models/sentence-transformers/all-MiniLM-L6-v2 HTTP/11" 200 6694
DEBUG:httpx:load_ssl_context verify=True cert=None trust_env=True http2=False
DEBUG:httpx:load_verify_locations cafile='/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/certifi/cacert.pem'
DEBUG:asyncio:Using selector: KQueueSelector
INFO: Started server process [25224]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:rag_system:Processing RAG for query: What is ROS2?
INFO:retrieval:Received query: What is ROS2?
Batches: 100% | 1/1 [00:00<00:00, 4.87it/s]
```

## rag\_gradio\_app.py

```
(Desktop/AI-FINAL) $ python3 rag_pipeline/rag_gradio_app.py
ClearML Task: overwriting (reusing) task id=17b4b13b1f284a119fc7f7b91f4a554d
2024-12-08 23:37:54,407 - clearml.Task - INFO - No repository found, storing script code instead
ClearML results page: https://app.clearml/projects/1f34e7ce98b8473bad3ae2da928d667d/experiments/17b4b13b1f284a119fc7f7b91f4a554d/output/log
ClearML only supports '0.0.0.0:7860' as the Gradio server. Ignoring name and port '0.0.0.0:7860' in remote execution
Running on local URL: http://0.0.0.0:7860

To create a public link, set 'share=True' in 'launch()'.
IMPORTANT: You are using gradio version 4.7.1, however version 4.44.1 is available, please upgrade.
-----
ClearML Monitor: GPU monitoring failed getting GPU reading, switching off GPU monitoring
Question asked: What is ROS2?
Q&A Pairs:

Final Answer:
I'm sorry, I couldn't generate an answer at this time.
Question asked: What is ROS2?
Q&A Pairs:
Q. What is ros2doctor?
A: "ros2doctor is a diagnostic tool for ROS 2 that checks the system configuration and identifies issues with the ROS 2 installation or runtime environment."

Q. What is ros2_control?
A: ros2_control is a flexible framework for real-time control of robots implemented using ROS 2, enabling smooth integration and control of hardware.

Q. What is a ROS 2 package?
A: A ROS 2 package is an organizational unit for your ROS 2 code that allows you to install, share, and easily build your code.

Q. What is a ROS 2 package?
A: An organizational unit for ROS 2 code, allowing it to be shared, installed, and easily used by others.

Q. What is the ros2 tool?
```

- **Evaluation of RAG System:**
  - Discuss how well the system performs in terms of answering domain-specific questions.
  - Include any evaluation metrics (e.g., accuracy, precision, recall, or qualitative evaluation).
  - Explain how you measure the utility and specificity of the answers provided by the system to ROS2 developers.
- **Challenges Faced:**
  - Discuss any technical challenges faced during the project (e.g., setting up Docker containers, fine-tuning the model, integration issues).
  - Mention the absence of OpenAI API key and how you worked around this limitation.

## 6. Conclusion and Future Work

- **Summary:** Summarize the work done and the overall success of building a RAG-based question-answering system for ROS2 robotics.
  - **Future Enhancements:**
    - Suggest areas for future improvement, such as better fine-tuning of the model, additional data sources, or more refined question-answer pair generation.
- 

## 7. References

- Include any references to academic papers, documentation, or resources you used throughout the project (e.g., Hugging Face documentation, ROS2 tutorials, ClearML, etc.).
- 

## Appendices

- Include any relevant code snippets, configuration files, or additional screenshots that may help to clarify the report.

## Member Details

**Pratham Saraf** – ps5218

**Siddh Mandirwala** – sm12505