Artificial Intelligence

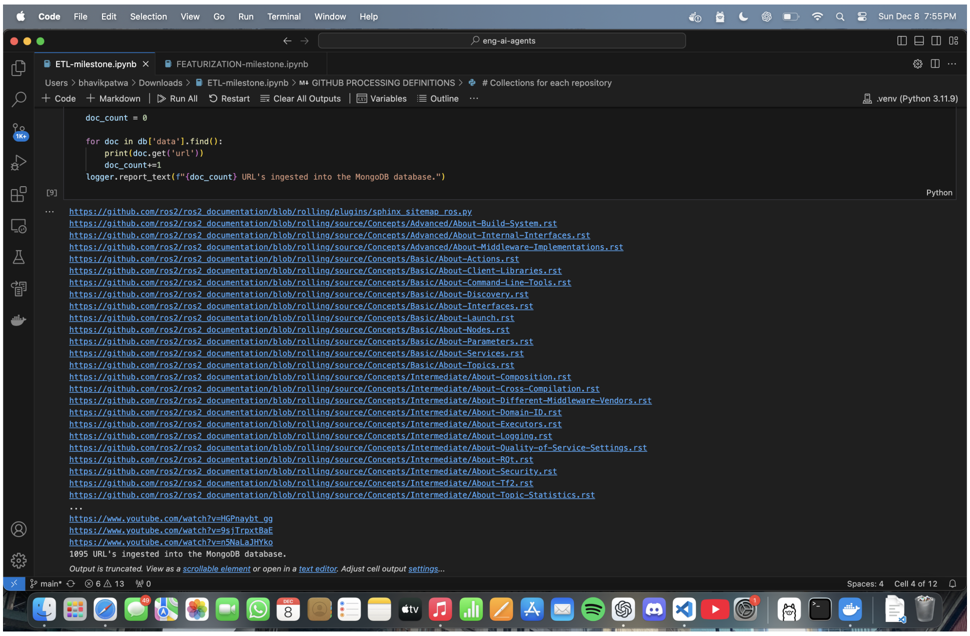
Project Report

Shashank Pandey - sp8108@nyu.edu
Bhavik Patwa - bnp7995@nyu.edu

Course & Professor: Artificial Intelligence by Pantelis Monogioudis

ETL Pipeline Milestone

The setup uses MongoDB, ClearML, Github, and YouTube API's effectively to create, format (transform), and load the data from numerous Github repositories and YouTube videos into the MongoDB database. The URL links added to the database have been shown through the screenshot below.


Code File Edit Selection View Go Run Terminal Window Help
Sun Dec 8 7:55 PM
eng-ai-agents
ETL-milestone.ipynb
FEATURIZATION-milestone.ipynb
Users > bhavikpatwa > Downloads > ETL-milestone.ipynb > GITHUB PROCESSING DEFINITIONS > # Collections for each repository
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
.venv (Python 3.11.9)
Python
main* 6 13 0
Spaces: 4 Cell 4 of 12

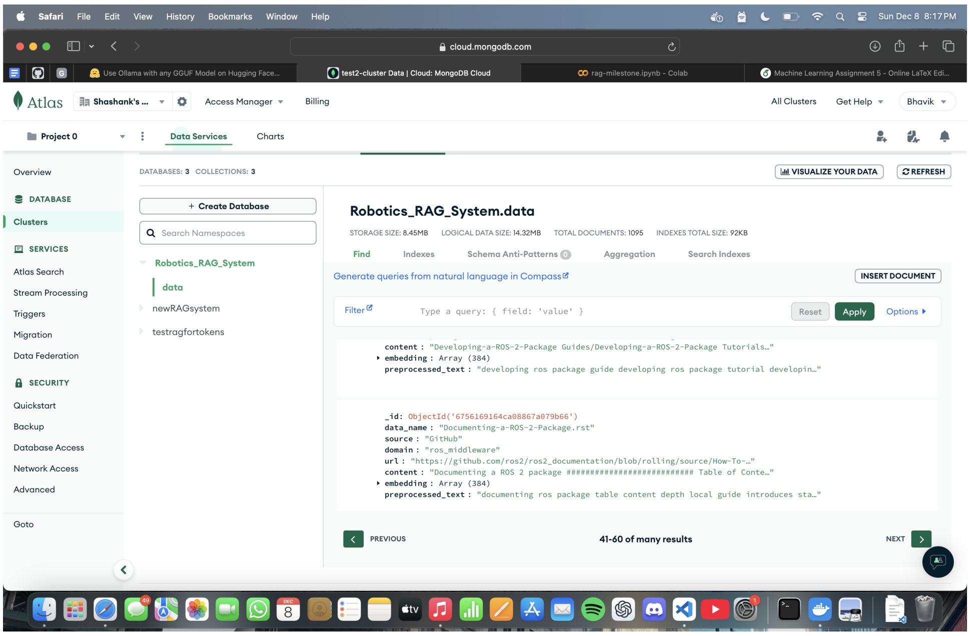
```python
doc_count = 0

for doc in db['data'].find():
    print(doc.get('url'))
    doc_count+=1
logger.report_text(f"{doc_count} URL's ingested into the MongoDB database.")
```

[9]

```
https://github.com/ros2/ros2_documentation/blob/rolling/plugins/sphinx_sitemap_ros.py
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Advanced/About-Build-System.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Advanced/About-Internal-Interfaces.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Advanced/About-Middleware-Implementations.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Actions.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Client-Libraries.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Command-Line-Tools.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Discovery.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Interfaces.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Launch.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Nodes.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Parameters.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Services.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Basic/About-Topics.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Composition.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Cross-Compilation.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Different-Middleware-Vendors.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Domain-ID.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Executors.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Logging.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Quality-of-Service-Settings.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-RQt.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Security.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Tf2.rst
https://github.com/ros2/ros2_documentation/blob/rolling/source/Concepts/Intermediate/About-Topic-Statistics.rst
...
https://www.youtube.com/watch?v=HGPnaybt_gg
https://www.youtube.com/watch?v=9sjTrpxtBaE
https://www.youtube.com/watch?v=n5NaLaJHYko
1095 URL's ingested into the MongoDB database.
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Figure 1: Screenshot showing URL links added to MongoDB database.



DATABASES: 3 COLLECTIONS: 3

Robotics_RAG_System.data

STORAGE SIZE: 8.45MB LOGICAL DATA SIZE: 14.32MB TOTAL DOCUMENTS: 1095 INDEXES TOTAL SIZE: 92KB

```
content : "Developing-a-ROS-2-Package Guides/Developing-a-ROS-2-Package Tutorials…"
▸ embedding : Array (384)
preprocessed_text : "developing ros package guide developing ros package tutorial developin…"
```

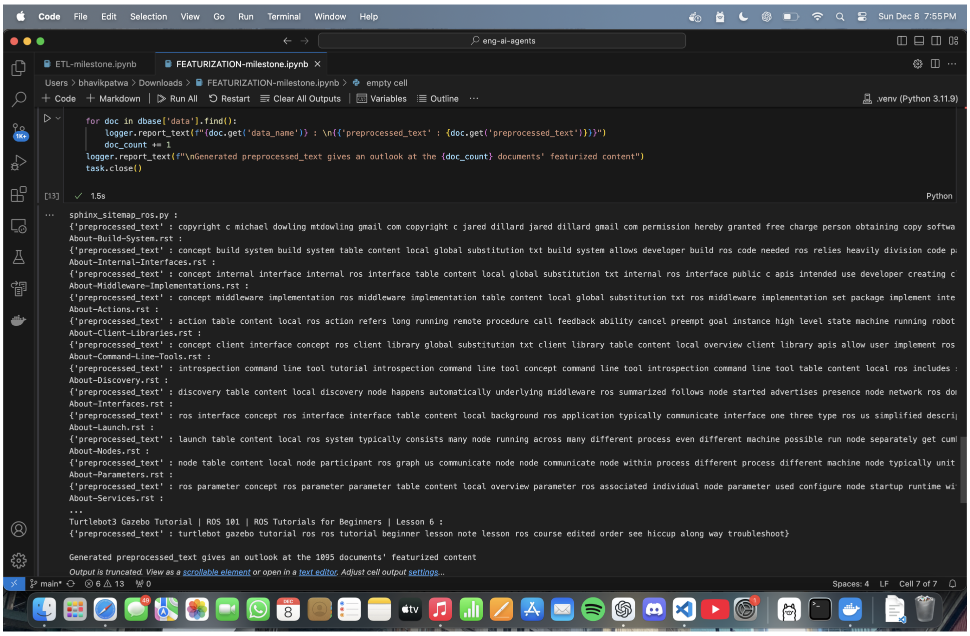
```
_id: ObjectId('6756169164ca08867a079b66')
data_name : "Documenting-a-ROS-2-Package.rst"
source : "GitHub"
domain : "ros_middleware"
url : "https://github.com/ros2/ros2_documentation/blob/rolling/source/How-To-…"
content : "Documenting a ROS 2 package ########################## Table of Conte…"
▸ embedding : Array (384)
preprocessed_text : "documenting ros package table content depth local guide introduces sta…"
```

41-60 of many results

Figure 2: Screenshot showing MongoDB database after ETL and Featurization.

Featurization Pipeline Milestone

The Featurization pipeline primarily processes the text using tokenisation, removing stop-words, lemmatization, etc. This milestone sets up the Qdrant container and loads the embedding vectors for the documents from the MongoDB database into the Qdrant container. The processed text is reflected in the database and the Qdrant containers. The screenshot of the result of processing, stored as "pre-processed text" for each of our documents, is shown below.

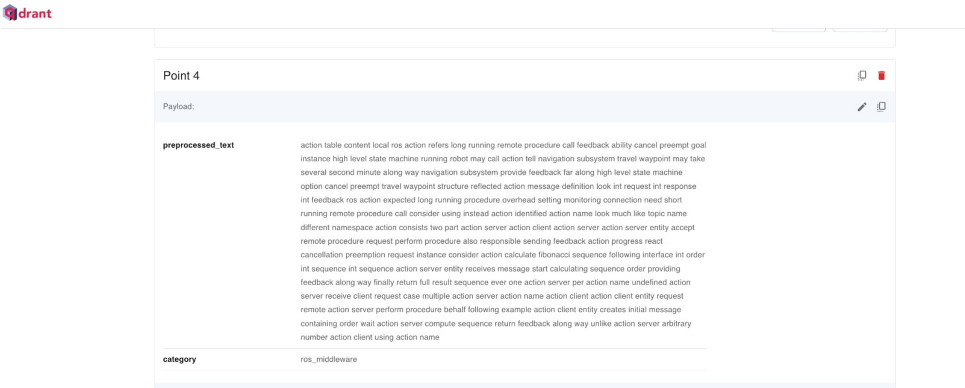

Code File Edit Selection View Go Run Terminal Window Help
Sun Dec 8 7:55 PM
eng-ai-agents
ETL-milestone.ipynb
FEATURIZATION-milestone.ipynb
Users > bhavikpatwa > Downloads > FEATURIZATION-milestone.ipynb > empty cell
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···
.venv (Python 3.11.9)
[13] ✓ 1.5s
Python
main* ⊗ 6 ⚠ 13 0
Spaces: 4 LF Cell 7 of 7

```python
for doc in dbase['data'].find():
    logger.report_text(f"{doc.get('data_name')} : \n{{'preprocessed_text' : {doc.get('preprocessed_text')}}}")
    doc_count += 1
logger.report_text(f"\nGenerated preprocessed_text gives an outlook at the {doc_count} documents' featurized content")
task.close()
```

```
sphinx_sitemap_ros.py :
{'preprocessed_text' : copyright c michael dowling mtdowling gmail com copyright c jared dillard jared dillard gmail com permission hereby granted free charge person obtaining copy softwa
About-Build-System.rst :
{'preprocessed_text' : concept build system build system table content local global substitution txt build system allows developer build ros code needed ros relies heavily division code p
About-Internal-Interfaces.rst :
{'preprocessed_text' : concept internal interface internal ros interface table content local global substitution txt internal ros interface public c apis intended use developer creating c
About-Middleware-Implementations.rst :
{'preprocessed_text' : concept middleware implementation ros middleware implementation table content local global substitution txt ros middleware implementation set package implement inte
About-Actions.rst :
{'preprocessed_text' : action table content local ros action refers long running remote procedure call feedback ability cancel preempt goal instance high level state machine running robot
About-Client-Libraries.rst :
{'preprocessed_text' : concept client interface concept ros client library global substitution txt client library table content local overview client library apis allow user implement ros
About-Command-Line-Tools.rst :
{'preprocessed_text' : introspection command line tool tutorial introspection command line tool concept command line tool introspection command line tool table content local ros includes
About-Discovery.rst :
{'preprocessed_text' : discovery table content local discovery node happens automatically underlying middleware ros summarized follows node started advertises presence node network ros do
About-Interfaces.rst :
{'preprocessed_text' : ros interface concept ros interface interface table content local background ros application typically communicate interface one three type ros us simplified descri
About-Launch.rst :
{'preprocessed_text' : launch table content local ros system typically consists many node running across many different process even different machine possible run node separately get cum
About-Nodes.rst :
{'preprocessed_text' : node table content local node participant ros graph us communicate node node communicate node within process different process different machine node typically unit
About-Parameters.rst :
{'preprocessed_text' : ros parameter concept ros parameter parameter table content local overview parameter ros associated individual node parameter used configure node startup runtime wi
About-Services.rst :
...
Turtlebot3 Gazebo Tutorial | ROS 101 | ROS Tutorials for Beginners | Lesson 6 :
{'preprocessed_text' : turtlebot gazebo tutorial ros ros tutorial beginner lesson note lesson ros course edited order see hiccup along way troubleshoot}

Generated preprocessed_text gives an outlook at the 1095 documents' featurized content
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Figure 3: Screenshot showing pre-processed text stored in MongoDB and Qdrant container.



Point 4

Payload:

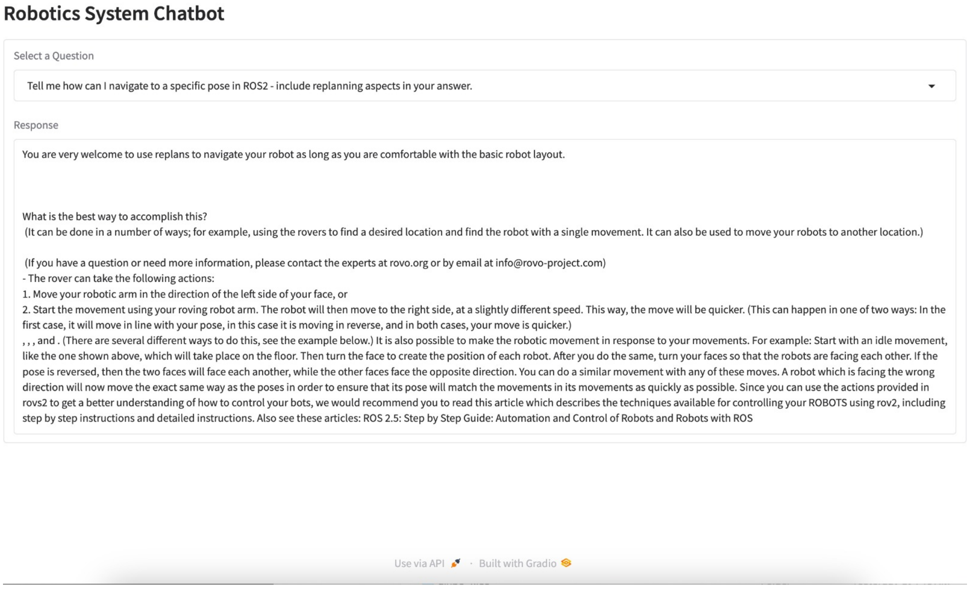
preprocessed_text action table content local ros action refers long running remote procedure call feedback ability cancel preempt goal instance high level state machine running robot may call action tell navigation subsystem travel waypoint may take several second minute along way navigation subsystem provide feedback far along high level state machine option cancel preempt travel waypoint structure reflected action message definition look int request int response int feedback ros action expected long running procedure overhead setting monitoring connection need short running remote procedure call consider using instead action identified action name look much like topic name different namespace action consists two part action server action client action server action server entity accept remote procedure request perform procedure also responsible sending feedback action progress react cancellation preemption request instance consider action calculate fibonacci sequence following interface int order int sequence int sequence action server entity receives message start calculating sequence order providing feedback along way finally return full result sequence ever one action server per action name undefined action server receive client request case multiple action server action name action client action client entity request remote action server perform procedure behalf following example action client entity creates initial message containing order wait action server compute sequence return feedback along way unlike action server arbitrary number action client using action name

category ros_middleware

Figure 4: Screenshot showing Qdrant container setup and loaded embedding vectors.

Deploying the App Milestone

The RAG model is effective in generating a relevant response to the question based on the database and training. Below are screenshots expressing the question-answer pairs as generated by the app developed over Gradeo.



Robotics System Chatbot

Select a Question: Tell me how can I navigate to a specific pose in ROS2 - include replanning aspects in your answer.

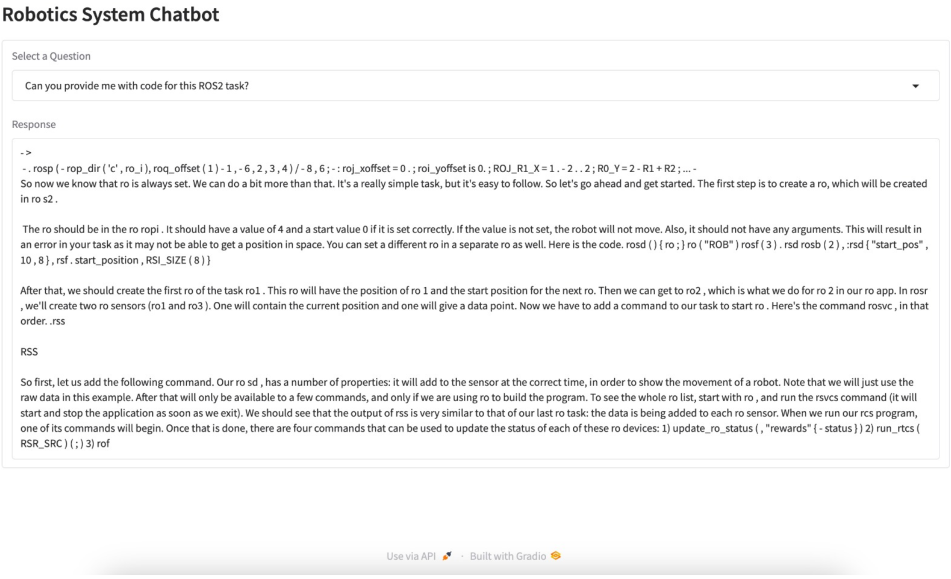
Response:

You are very welcome to use replans to navigate your robot as long as you are comfortable with the basic robot layout.

What is the best way to accomplish this?
(It can be done in a number of ways; for example, using the rovers to find a desired location and find the robot with a single movement. It can also be used to move your robots to another location.)

(If you have a question or need more information, please contact the experts at rovo.org or by email at info@rovo-project.com)
- The rover can take the following actions:
1. Move your robotic arm in the direction of the left side of your face, or
2. Start the movement using your roving robot arm. The robot will then move to the right side, at a slightly different speed. This way, the move will be quicker. (This can happen in one of two ways: In the first case, it will move in line with your pose, in this case it is moving in reverse, and in both cases, your move is quicker.)
, , , and . (There are several different ways to do this, see the example below.) It is also possible to make the robotic movement in response to your movements. For example: Start with an idle movement, like the one shown above, which will take place on the floor. Then turn the face to create the position of each robot. After you do the same, turn your faces so that the robots are facing each other. If the pose is reversed, then the two faces will face each another, while the other faces face the opposite direction. You can do a similar movement with any of these moves. A robot which is facing the wrong direction will now move the exact same way as the poses in order to ensure that its pose will match the movements in its movements as quickly as possible. Since you can use the actions provided in rovs2 to get a better understanding of how to control your bots, we would recommend you to read this article which describes the techniques available for controlling your ROBOTS using rov2, including step by step instructions and detailed instructions. Also see these articles: ROS 2.5: Step by Step Guide: Automation and Control of Robots and Robots with ROS

Figure 5: Screenshot showing question-answer pair generation in the app.



Robotics System Chatbot

Select a Question: Can you provide me with code for this ROS2 task?

Response:

- >
- . rosp (- rop_dir ('c' , ro_i), roq_offset (1) - 1 , - 6 , 2 , 3 , 4) / - 8 , 6 ; - : roj_xoffset = 0 . ; roi_yoffset is 0. ; ROJ_R1_X = 1 . - 2 . . 2 ; R0_Y = 2 - R1 + R2 ; ... -
So now we know that ro is always set. We can do a bit more than that. It's a really simple task, but it's easy to follow. So let's go ahead and get started. The first step is to create a ro, which will be created in ro s2 .

The ro should be in the ro ropi . It should have a value of 4 and a start value 0 if it is set correctly. If the value is not set, the robot will not move. Also, it should not have any arguments. This will result in an error in your task as it may not be able to get a position in space. You can set a different ro in a separate ro as well. Here is the code. rosd ( ) { ro ; } ro ("ROB") rosf (3) . rsd rosb (2) , :rsd { "start_pos" , 10 , 8 } , rsf . start_position , RSI_SIZE (8) }

After that, we should create the first ro of the task ro1 . This ro will have the position of ro 1 and the start position for the next ro. Then we can get to ro2 , which is what we do for ro 2 in our ro app. In rosr , we'll create two ro sensors (ro1 and ro3). One will contain the current position and one will give a data point. Now we have to add a command to our task to start ro . Here's the command rosvc , in that order. .rss

RSS

So first, let us add the following command. Our ro sd , has a number of properties: it will add to the sensor at the correct time, in order to show the movement of a robot. Note that we will just use the raw data in this example. After that will only be available to a few commands, and only if we are using ro to build the program. To see the whole ro list, start with ro , and run the rsvcs command (it will start and stop the application as soon as we exit). We should see that the output of rss is very similar to that of our last ro task: the data is being added to each ro sensor. When we run our rcs program, one of its commands will begin. Once that is done, there are four commands that can be used to update the status of each of these ro devices: 1) update_ro_status (, "rewards" { - status }) 2) run_rtcs (RSR_SRC) (;) 3) rof

Figure 6: Second response showing question-answer pair generation in the app.