

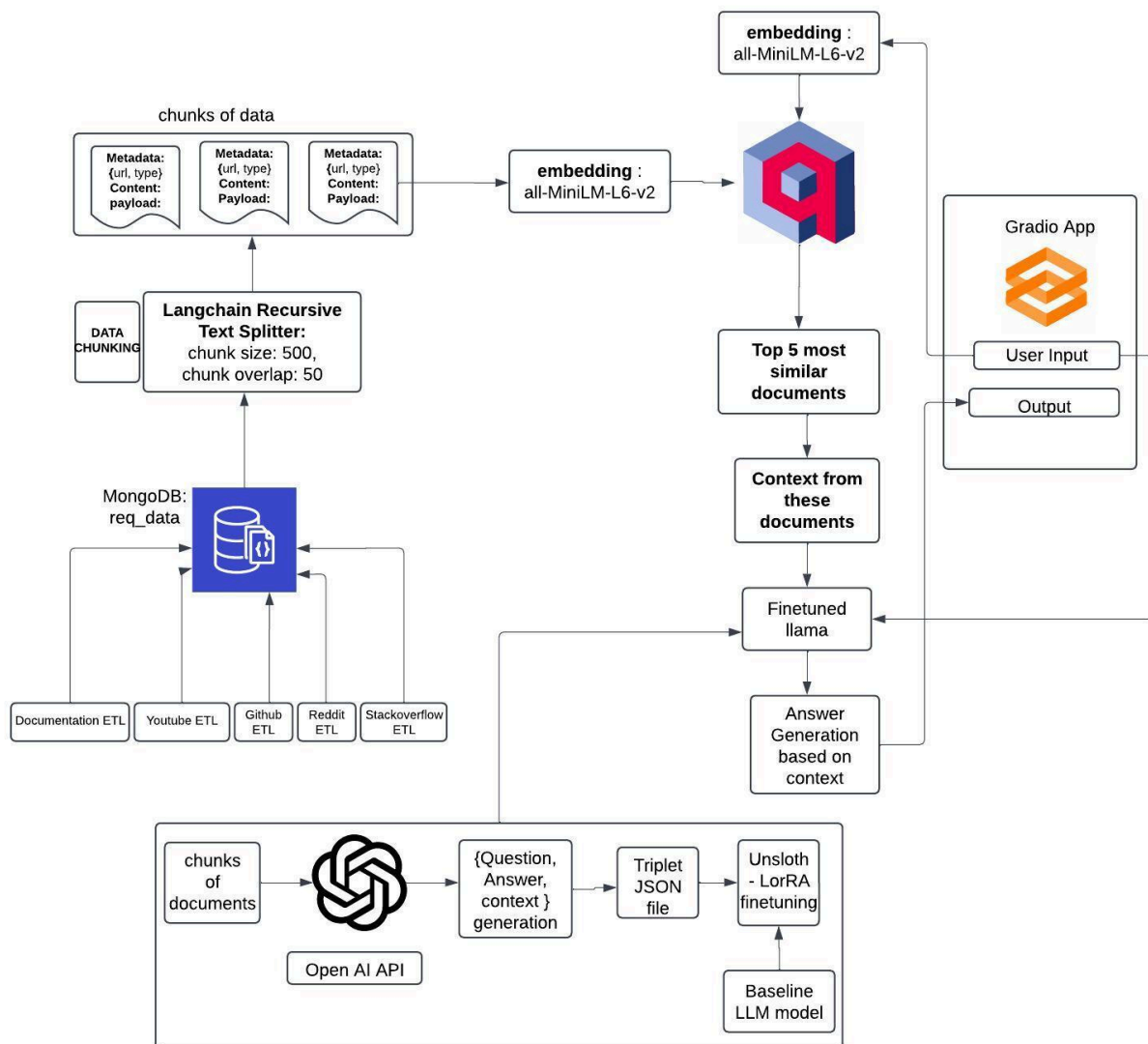
Project Report: Fine Tuned RAG systems engineerings

Yashavika Singh and Debika Piriya Dharma Lingam
CS-GY-6613

Contents

Contents	2
Architecture	3
MongoDB Collections:	4
Docker Setup	5
ETL Pipeline	6
Medium ETL	7
Reddit ETL	8
Stackoverflow ETL	9
Documentation ETL	10
Youtube ETL	11
Chunking using Langchain RecursiveCharacterTextSplitter	12
Creating embeddings	12
Pushing chunks to Qdrant	12
Gradio App	13
User input	13
Retrieving from Qdrant	13
LLM	13
Displaying Output	13
Fine tuning the LLM	14
Creating the Question, Answer, Context triplets	14
Training the model	15
Pushing the trained model to Huggingface	15
zenML Pipeline	16

Architecture



MongoDB Collections:

localhost:27017 > req_data

Sort by

Collection Name

1F

gazebo_documentation

<div>Storage size:</div> <div>245.76 kB</div>	<div>Documents:</div> <div>106</div>	<div>Avg. document size:</div> <div>6.48 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>36.86 kB</div>
---	--------------------------------------	---	----------------------------------	--

medium

<div>Storage size:</div> <div>24.58 kB</div>	<div>Documents:</div> <div>9</div>	<div>Avg. document size:</div> <div>2.27 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>36.86 kB</div>
--	------------------------------------	---	----------------------------------	--

moveit_documentation

<div>Storage size:</div> <div>90.11 kB</div>	<div>Documents:</div> <div>55</div>	<div>Avg. document size:</div> <div>3.83 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>36.86 kB</div>
--	-------------------------------------	---	----------------------------------	--

nav2_documentation

<div>Storage size:</div> <div>8.43 MB</div>	<div>Documents:</div> <div>769</div>	<div>Avg. document size:</div> <div>29.62 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>53.25 kB</div>
---	--------------------------------------	--	----------------------------------	--

reddit

<div>Storage size:</div> <div>65.54 kB</div>	<div>Documents:</div> <div>72</div>	<div>Avg. document size:</div> <div>1.39 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>20.48 kB</div>
--	-------------------------------------	---	----------------------------------	--

ros2_documentation

<div>Storage size:</div> <div>1.34 MB</div>	<div>Documents:</div> <div>253</div>	<div>Avg. document size:</div> <div>14.66 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>36.86 kB</div>
---	--------------------------------------	--	----------------------------------	--

stackoverflow

<div>Storage size:</div> <div>20.48 kB</div>	<div>Documents:</div> <div>3</div>	<div>Avg. document size:</div> <div>1.48 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>20.48 kB</div>
--	------------------------------------	---	----------------------------------	--

youtube_captions

<div>Storage size:</div> <div>135.17 kB</div>	<div>Documents:</div> <div>56</div>	<div>Avg. document size:</div> <div>6.15 kB</div>	<div>Indexes:</div> <div>1</div>	<div>Total index size:</div> <div>20.48 kB</div>
---	-------------------------------------	---	----------------------------------	--

Docker Setup

Screenshot of docker ps command, indicating all services are working

```
Last login: Wed Nov 20 21:31:07 on ttys022
(base) yashavikasingh@Yashoos-MacBook-Air RAG % docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                                     NAMES
ab312137acca   python:3.10-slim                   "bash -c ' pip insta..." 11 minutes ago Up 7 seconds  0.0.0.0:5001->5000/tcp                  app
5b9413daa38b   allegroai/clearml:latest           "/opt/clearml/wrappe..." 22 minutes ago Up 7 seconds  0.0.0.0:8008->8008/tcp, 0.0.0.0:8080-8081->8080-8081/tcp clearml
ec7ddc789252   mongo:6.0                           "docker-entrypoint.s..." 22 minutes ago Up 7 seconds  0.0.0.0:27017->27017/tcp                mongod
0181b85dc604   qdrant/qdrant:v1.3.0               "./entrypoint.sh"         22 minutes ago Up 7 seconds  0.0.0.0:6333->6333/tcp, 6334/tcp        qdrant
(base) yashavikasingh@Yashoos-MacBook-Air RAG %
```

```
[debikad@Debikas-MacBook-Pro AI-project % docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                                     NAMES
49da1e46aeecc ai-project-app                       "python app.py"         2 days ago    Up 2 days    0.0.0.0:5001->5000/tcp                  app
e5f8fa1dbd777 mongo:5.0                           "docker-entrypoint.s..." 2 days ago    Up 2 days    0.0.0.0:27017->27017/tcp                mongod
6d856102389e   qdrant/qdrant:v1.3.0               "docker-entrypoint.s..." 2 days ago    Up 2 days    0.0.0.0:6333->6333/tcp, 6334/tcp        qdrant
bacab7d7e398   allegroai/clearml:latest           "/opt/clearml/wrappe..." 2 days ago    Up 9 seconds  0.0.0.0:8008->8008/tcp, 0.0.0.0:8080-8081->8080-8081/tcp clearml
```

Containers [Give feedback](#)

Container CPU usage ⓘ

0.86% / 800% (8 CPUs available)

Container memory usage ⓘ

784.62MB / 3.74GB

[Show charts](#)

🔍 Search

🔧

Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	suspicious_shockley	35a785ec0a68	qdrant/qdrant	6333:6333	0%		<div><div>▶</div><div>⋮</div><div>🗑</div></div>
<input type="checkbox"/>	ai-project	-	-	-	0.86%	4 days ago	<div><div>☐</div><div>⋮</div><div>🗑</div></div>
<input type="checkbox"/>	app-1	0c32b61dbc06	ai-project-app	8000:8000 🔗	0.03%	4 days ago	<div><div>☐</div><div>⋮</div><div>🗑</div></div>
<input type="checkbox"/>	mongodb	f27b10794d97	mongo:5.0	27017:27017 🔗	0%	4 days ago	<div><div>☐</div><div>⋮</div><div>🗑</div></div>
<input type="checkbox"/>	clearml	8c2408d8034c	allegroai/clearml:late	8080:8080	0%	4 days ago	<div><div>▶</div><div>⋮</div><div>🗑</div></div>
<input type="checkbox"/>	qdrant	fcda01591de3	qdrant/qdrant:v1.3.0	6333:6333 🔗	0.83%	4 days ago	<div><div>☐</div><div>⋮</div><div>🗑</div></div>
<input type="checkbox"/>	eng-ai-agents	-	-	-	0%	20 days ago	<div><div>▶</div><div>⋮</div><div>🗑</div></div>

ETL Pipeline

Data Sources used:

1. Reddit
2. Medium
3. StackOverflow
4. Documentation: ros2, nav2, gazebo, moveit
5. Youtube

Medium ETL

We take a list of medium websites, extract HTML content by using the requests library and sending a GET request. We removed the whitespace and non-printable characters. The metadata is extracted using the BeautifulSoup library.

```
{  "metadata": {
    "type": "Medium",
    "url": link,
    "title": clean_text(title),
    "author": clean_text(author_name),
    "publication_date": clean_text(publication_date),
  },
  "content": article_content
}
```

We ingested the following 3 articles:

```
medium_links = [
    "https://medium.com/schmiedeone/getting-started-with-ros2-part-1-d4c3b7335c71",
    "https://medium.com/@nullbyte.in/ros2-from-the-ground-up-part-1-an-introduction-to-the-robot-operating-system-4c2065c5e032",
    "https://medium.com/@tetraengnrng/a-beginners-guide-to-ros2-29721dcf49c8"
]
```

Reddit ETL

I first created an app in Reddit to get the client id, client secret and user agent. The praw library is used to get a limited number of posts that contain a keyword, from a given subreddit. The clean function removes whitespaces, removes urls, and non-printable characters.

We get 10 posts each from each of the following subreddit and keyword combinations:

```
configurations = [  
    {"subreddit": "ROS", "keyword": "ROS2"},  
    {"subreddit": "ROS", "keyword": "nav2"},  
    {"subreddit": "ROS", "keyword": "gazebo"},  
    {"subreddit": "ROS", "keyword": "moveit"},  
]
```

It is stored in this format.

```
{  
  "metadata":  
  {  
    "type": "reddit",  
    "subreddit": subreddit,  
    "keyword": keyword,  
    "url": f"https://reddit.com{post.permalink}",  
  },  
  "content": content  
}
```


Stackoverflow ETL

We extracted data from a list of stackoverflow urls. We used the requests library to fetch data from the urls. The clean function removes html tags and whitespace. We used the Stack Exchange API to get the responses. We inputted the following urls:

```
urls = [  
    "https://stackoverflow.com/questions/57426715/import-modules-in-package-in-ros2",  
    "https://stackoverflow.com/questions/51187676/whats-the-difference-between-ros2-and-dds",  
  
    "https://stackoverflow.com/questions/68771051/ros2-pub-sub-custom-message-through-ros2-web-bridge-to-client-app",  
]
```

We store it in this format in mongodb.

```
{  
  {"metadata": {  
    "type": "Stackoverflow",  
    "url": url,  
    "title": question_data.get("title", "Untitled")  
  },  
  "content": content  
}
```

Documentation ETL

Selenium and BeautifulSoup was used

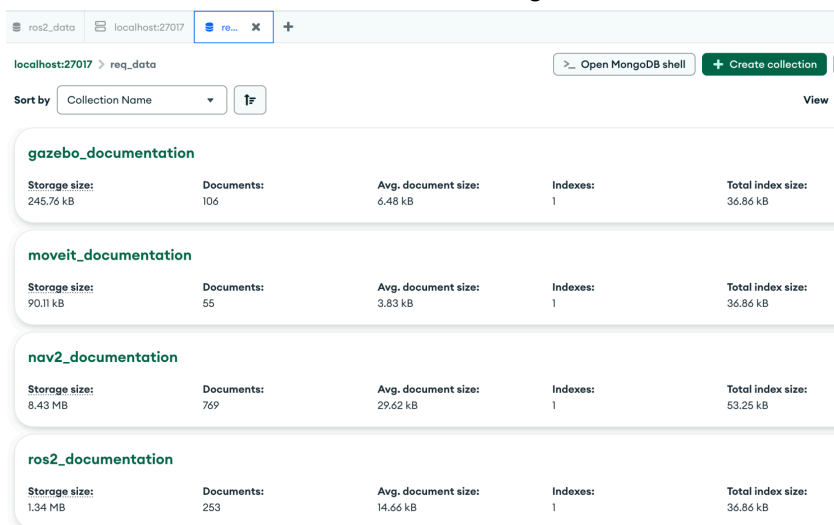
After the content is extracted, we clean the text by removing: white space, non printable characters, urls.

We first save the base page content, then from the base page, we scrape links to other pages, and then we go to that list of links to pages and scrape links and content from those pages.

The web pages from the following pages were scraped:

1. Ros2 documentation: <https://docs.ros.org/en/foxy/index.html>
 - 253 pages were scraped
2. Nav2 documentation: <https://docs.nav2.org/>
 - 300 pages were scraped using keywords "navigation", "path planning", "setup", "ROS", "map", "robot"
3. Gazebo documentation: <https://gazebo.org/home>
 - 106 pages were scraped
4. Moveit documentation: <https://moveit.ai/>
 - 55 pages were scraped

A different collection was created in mongodb for each of these websites



Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
gazebo_documentation	245.76 kB	106	6.48 kB	1	36.86 kB
moveit_documentation	90.11 kB	55	3.83 kB	1	36.86 kB
nav2_documentation	8.43 MB	769	29.62 kB	1	53.25 kB
ros2_documentation	1.34 MB	253	14.66 kB	1	36.86 kB

Youtube ETL

I created a list of youtube video ids by passing in a list of playlist urls. Then for each video id, I used the third party API: YoutubeTranscriptAPI to get captions. The youtube API was very restrictive and to bypass the quota, I used the YoutubeTranscriptAPI instead to get the captions. I still use the youtube API to get video details: the name of the video and the title. I used the OAuth authentication service. I stored my OAuth credentials in a json file and then loaded it in a pickle file for reuse.

The captions are cleaned by removing html tags and removing whitespace. Transform data function gets the cleaned captions, url and stores them in this format.

```
metadata = {
    "type": "YouTube",
    "url": url,
    "title": video_details["title"],
    "description": video_details["description"],
}
```

The data is saved in a collection called youtube_captions in mongodb, the following playlist was ingested: ROS2 Tutorials Humble by Kevin Wood:

https://www.youtube.com/watch?v=C6eQ6VwTpxk&list=PLSK7NtBWwmpTS_YVfieN3ZzIxtl1P_Sr

The screenshot shows the MongoDB Compass interface. On the left, the 'CONNECTIONS' panel lists the local host 'localhost:27017' with a tree view of databases including 'req_data' and 'youtube_captions'. The main panel displays the 'youtube_captions' collection with 56 documents. The 'Documents' tab is active, showing a list of documents. Each document has a unique '_id' and a 'metadata' object containing 'type', 'url', 'title', 'description', and 'content'. The documents represent video captions from a YouTube playlist.

_id	metadata.type	metadata.url	metadata.title	metadata.description	metadata.content
ObjectID('6754bc3c6c05977656647e6a')	YouTube	https://www.youtube.com/watch?v=C6eQ6VwTpxk	ROS2 Using VS Code and WSL	Code and Doc: https://kevinwoodrobotics.com/product/ros2-tutorials-ful-	all right in this video I'm going to show you how to set up WSL on Win.
ObjectID('6754bc3d6c05977656647e6c')	YouTube	https://www.youtube.com/watch?v=7FKf-waQu8M	ROS2 Setting Up Github In WSL	In this ROS2 tutorial video, I will go over how to setup Github in WSL.	all right in this video I'm going to show you how to set up Github in .
ObjectID('6754bc3e6c05977656647e6e')	YouTube	https://www.youtube.com/watch?v=d3LBb8IXdw	ROS2 Humble Installation	In this ROS2 tutorial video, I will go over how to install humble on w.	all right in this video I'm going to show you how to install Ross 2 hu.
ObjectID('6754bc3f6c05977656647e70')	YouTube	https://www.youtube.com/watch?v=72a-wJ2k25A	ROS2 Command Not Found	In this ROS2 tutorial video, I will talk about the common message begi.	all right in this video I want to address a common problem that people.
ObjectID('6754bc406c05977656647e72')	YouTube	https://www.youtube.com/watch?v=sWd9p1lMz0	ROS2 Running Executables	In this ROS2 tutorial video, I will go over how to run ROS executables.	all right in this video I'm going to show you how to run executables i.

Chunking using Langchain RecursiveCharacterTextSplitter

We created a list of dictionaries, each dictionary is a document in a mongodb collection

List of dictionaries called documents:

```
{"content": doc["content"], "metadata": metadata}
```

Metadata has the url and type of source

We used chunk size=500 and overlap=50, to create a new list of dictionaries called `chunked_data`.

Note: The package used ensures meaningful paragraphs were stored as chunks indeed of fixed character length

Creating embeddings

Our `generate_embeddings` function uses `all-MiniLM-L6-v2` to create embeddings for the content of each chunk.

all-miniLM-L6-v2

`all-MiniLM-L6-v2` is a sentence transformer created by Microsoft and HuggingFace. It is based on MiniLM which is a light-weight version of BERT. For each input it outputs a 384 dimension vector. It uses a WordPiece tokenizer and 6-layer transformer encoder. Lastly, it does a mean pooling operation.

Pushing chunks to Qdrant

The chunks, embeddings of all resources are pushed to one Qdrant collection called `unified_collection`, where we store the embeddings, url, the payload of each point we store the content, type and url.

Gradio App

User input

We take the input from the user, it is a text field.

Retrieving from Qdrant

We first create the embedding for the input using all-MiniLM-L6-v2, then retrieve the top 5 similar chunks from Qdrant.

LLM

We then use llama2.7b to generate the answer given the context and the query.

Displaying Output

Fine tuning the LLM

Creating the Question, Answer, Context triplets

We created 1600 question, answer and context triplets using OpenAI. We created the chunks from the documentation itself: nav2, ros2, gazebo and moveit. We used the chunks that were created using langchain.

We took in 1000 random samples,

We used gpt-3.5-turbo

The following is a sample of the json file:

```
[ {
  "question": "How can one view camera feed images based on the given content?",
  "answer": "To view camera feed images, one can select a `sensor_msgs/msg/Image` or `sensor_msgs/msg/CompressedImage` topic to display.",
  "context": "interactions.2 Diagnostics: Filter and sort diagnostics messagesDisplay the status of seen nodes (i.e. stale, error, warn, or OK) from topics with adiagnostic_msgs/msg/DiagnosticArraydatatype in a running feed, and display the diagnostics data for a givendiagnostic_name/hardware_id.Reference thedocsfor more details.3 Image: View camera feed imagesSelect asensor_msgs/msg/Imageorsensor_msgs/msg/CompressedImagetopic to display.Reference thedocsfor more details.4 Log: View log messagesTo"
},
{
  "question": "What can be done using the \"Log\" functionality mentioned in the content?",
  "answer": "The \"Log\" functionality allows users to view log messages.",
  "context": "interactions.2 Diagnostics: Filter and sort diagnostics messagesDisplay the status of seen nodes (i.e. stale, error, warn, or OK) from topics with adiagnostic_msgs/msg/DiagnosticArraydatatype in a running feed, and display the diagnostics data for a givendiagnostic_name/hardware_id.Reference thedocsfor more details.3 Image: View camera feed imagesSelect asensor_msgs/msg/Imageorsensor_msgs/msg/CompressedImagetopic to display.Reference thedocsfor more details.4 Log: View log messagesTo"
},
]
```

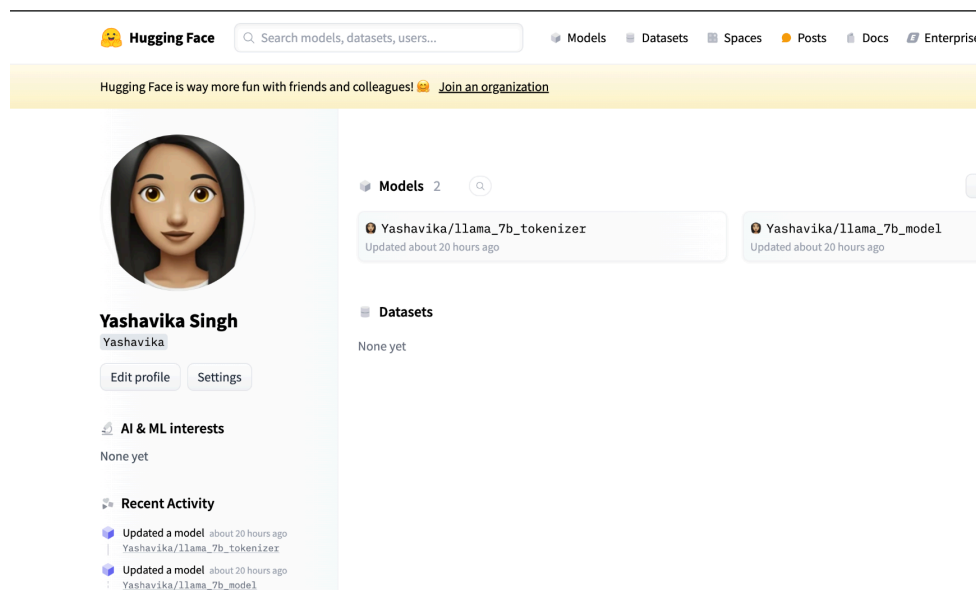
Training the model

We trained the model on google collab using the T4 GPU. We referenced the following article and it's code:

<https://medium.com/@sohanm10/a-step-by-step-guide-to-fine-tuning-llama-7b-with-unsloth-and-lora-bc00a90899a2>

Pushing the trained model to Huggingface

We pushed this trained model to huggingface.



zenML Pipeline

We used zenML since clearML was throwing errors. To create the pipeline we had to add type annotations to the functions and `@step` decorators to the function. The pipeline runs from the fetch document step to the store in qdrant step.

