CyberScienceLab

University of Guelph

Canada Cyber Foundry

Developing a RAG To Mitigate LLM Hallucinations in Threat Intelligence

Introduction

This project implements a system for managing Cyber Threat Intelligence by utilizing a Retrieval-Augmented Generation (RAG) model combined with Qdrant and Meta's LLaMA3. The objective is to validate the use of threat indicators extracted from threat intelligence platforms (like AlienVault OTX) and ensure that these indicators are contextually applied within intelligence reports. The system facilitates the extraction, storage, embedding, and querying of threat indicators, enabling efficient analysis and validation using advanced AI models.

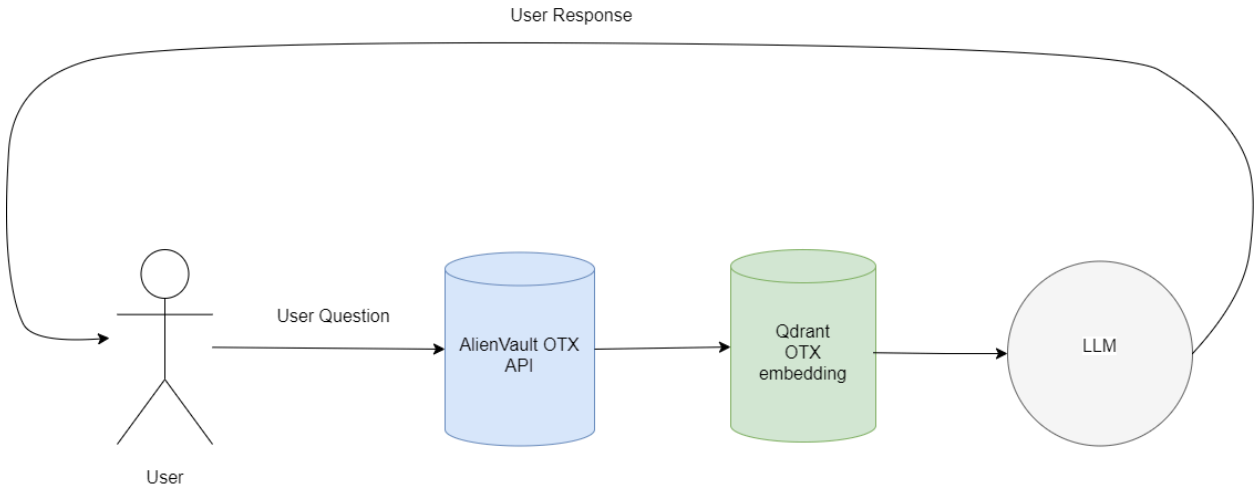


Diagram of Retrieval Augmented Generation with Cyber Threat Intelligence information

Upon receiving a user's question, the system retrieves relevant threat indicators (pulses) from the AlienVault OTX API. The information is processed and formatted into a structured format suitable for further analysis. Simultaneously, the system utilizes Qdrant embeddings to perform similarity searches, drawing from a vector space created from the processed data. The compiled information is then fed into LLaMA3, which analyzes and incorporates the threat intelligence to provide a comprehensive and contextually relevant response to the user.

System Overview

1. **Data Retrieval**: The system retrieves threat indicators (pulses) from the AlienVault OTX API.
2. **Data Processing**: The information is structured for analysis and similarity searches using Qdrant embeddings.
3. **Model Integration**: The processed data is fed into LLaMA3 to generate a contextually relevant response to user queries.

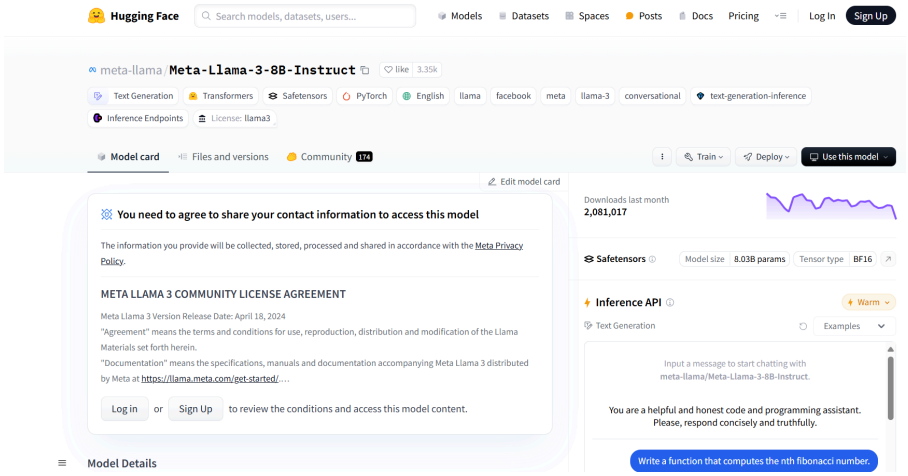
Step by step Demonstration

GITHUB ACCESS

1. **Preparing Qdrant and the environment:**
   Before embedding your data, set up the environment and the Qdrant Docker image by running the necessary command in your terminal. This will allow the Qdrant application to be up and running.
   1. Setup and activate project Anaconda environment
      > conda create --name ENV_NAME python=3.12
      > conda activate ENV_NAME
   2. Install all dependencies
      > pip install -r requirements.txt
   3. Start the Qdrant container
      > docker run -p 6333:6333 qdrant/qdrant
   4. Verify container started successfully and are running
      > docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6fafbeab618c	postgres	"docker-entrypoint.s…"	12 days ago	Up 12 days	0.0.0.0:5432->5432/tcp, :::5432->5432/tcp	rag_postgres
f9ad8200a51f	qdrant/qdrant	"./entrypoint.sh"	12 days ago	Up 7 days	0.0.0.0:6333->6333/tcp, :::6333->6333/tcp, 6334/tcp	inspiring_dewdney

Access Llama3 HuggingFace Token:
[HuggingFace website]



1. Create/Log into your Huggingface account
2. Get access to Llama3's Token
3. Export the token in your python environment by typing the command: export HF_TOKEN=<YOUR TOKEN>

Gather Data:

We'll be using AlienVault OTX as our primary data source. The first step is to OTX API KEY available on their website. To do this, visit [DirectConnect API] and click copy on "Your OTX Key " icon. **You will have to create a free account first.**

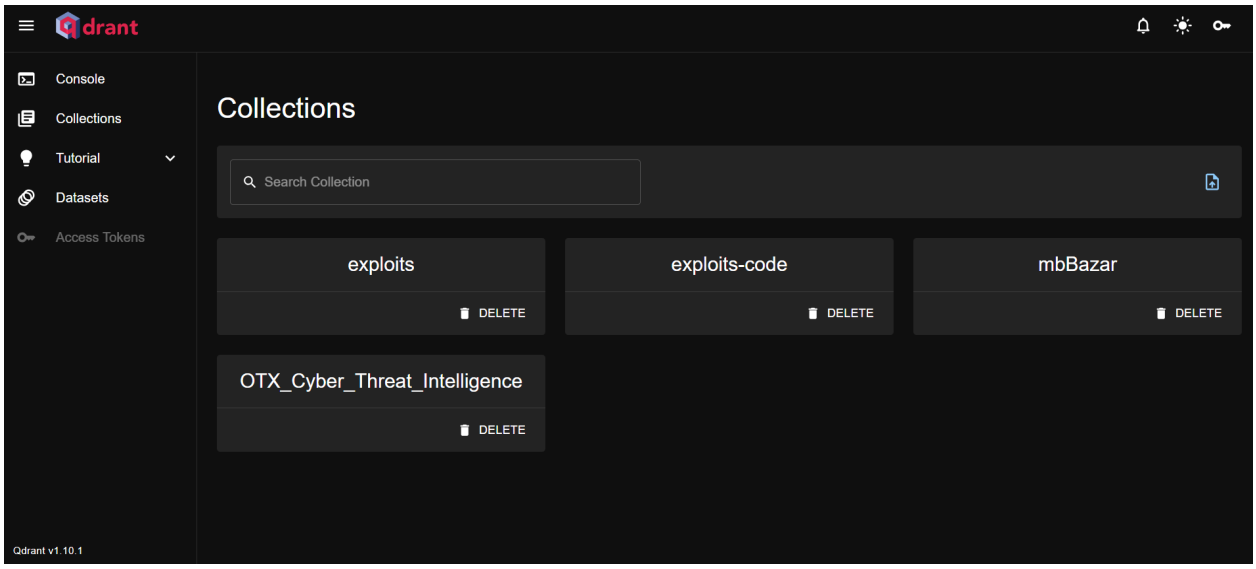
Gathering Data:

****For additional help and resources visit the official [OTX Github](#)****

1. **Run the python Script:**
   Execute the `Text_data_collection.py` script to begin the data extraction process.
   ```
   /Threat_Intelligence_Rag$ text_data_collection.py
   ```
2. **Creating the CSV Files:**
   The script will automatically open the `all_pulses_filtered.csv` file that was filtered and exported from AlienVault.
   It will be used to create a new csv called `indicators_extracted.csv` which takes the last column of `all_pulses_filtered.csv` which is in a json format so that it can be used as a csv.

Embedding Data:

1. **Checking Qdrant:**
   You can confirm that Qdrant is running by checking the Qdrant UI, which should be accessible at http://localhost:6333/collections.



2. **Set Up the Collection:**
   Open the `OTXrag.py` script in your code editor.

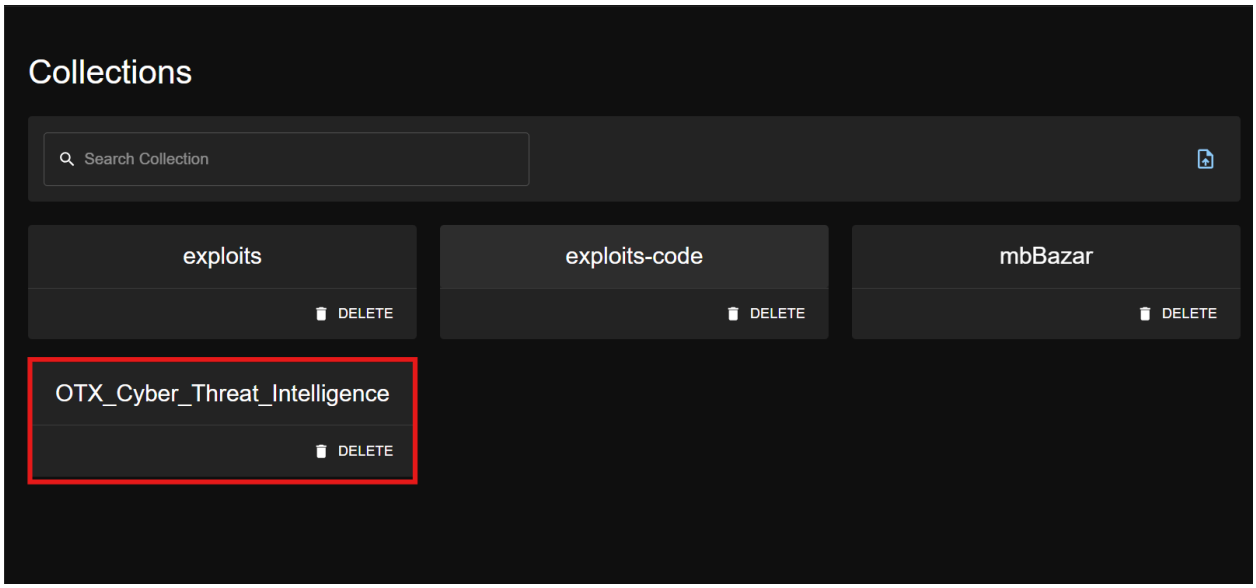
If the collection hasn't been named yet, assign it a name within the script. For this case, the collection has already been named `OTX_Cyber_Threat_Intelligence`.

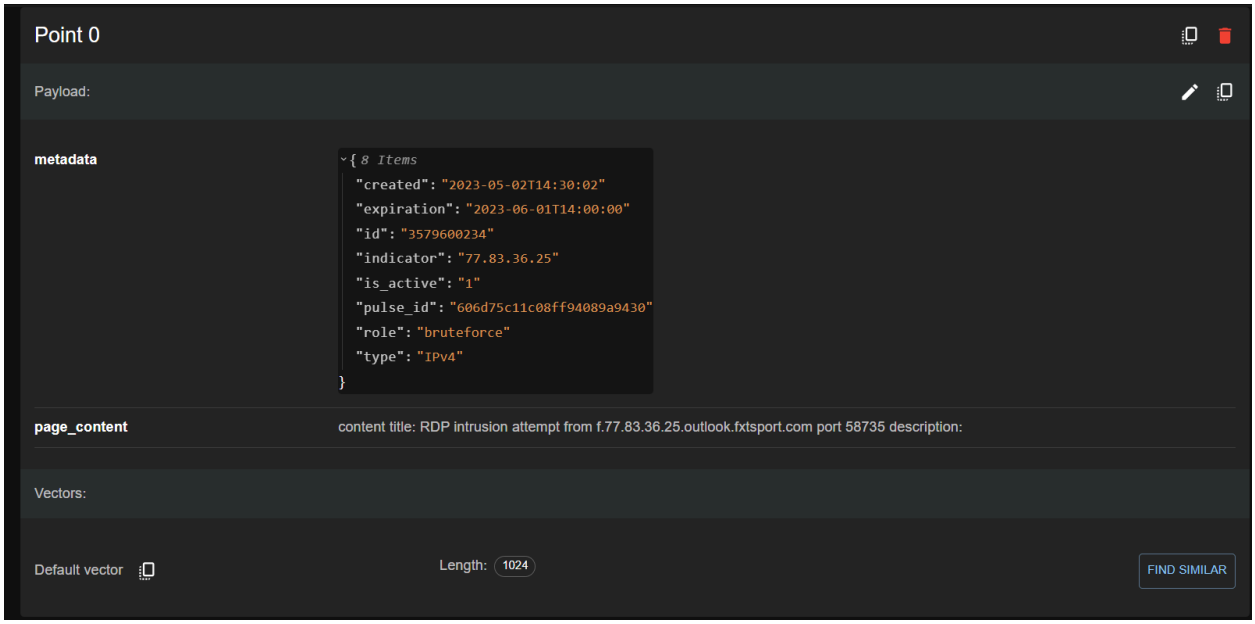
```
# Configuration constants for Qdrant and other settings.
QDRANT_URL = "http://localhost:6333"
DEFAULT_COLLECTION_NAME = "OTX_Cyber_Threat_Intelligence"
DEFAULT_VECTOR_SIZE = 1024
DEFAULT_CHUNK_SIZE = 1000
DEFAULT_CHUNK_OVERLAP = 200
BATCH_SIZE = 100
EMBEDDING_DELAY_SECONDS = 5
```

3. **Run the Embedding Script:**
   After making the necessary adjustments, run the script by entering the command to execute it. This will begin the process of embedding the data.

```
/Threat Intelligence Rag$ OTXrag.py
```

4. **Monitor Progress:**
   The embedding process can take some time, ranging from a few minutes to several hours, depending on the amount of data and hardware you are working with.
5. **Verify Embeddings:**
   Once the script has finished running, you should be able to see the embedded data by visiting the Qdrant UI at http://localhost:6333/collections.





To Learn More About running the RAG UI Program and accessing a Step-by-step demonstration Visit our [RAG_App Github](#).

University of Guelph

Canada Cyber Foundry

CyberScienceLab

Secure the future with us

cybersciencelab.com