RAG based Chatbot for Museum

In today's digital era, conversational AI technologies play a crucial role in enhancing user engagement and providing valuable information to users. In this report, I present the development of a question-answering (QA) chatbot designed specifically for a museum setting. The chatbot leverages retrieval-based techniques to provide accurate and informative responses to user queries regarding museum-related topics.

Objective

The primary objective of this project is to create a user-friendly and efficient QA chatbot that can assist visitors with inquiries related to the museum's exhibits, history, and services. The chatbot aims to enhance the overall visitor experience by providing prompt and accurate responses to a wide range of questions.

Implementation

Data Collection

Web scraping is a technique used to extract data or content from websites for various purposes, such as data analysis, research, or archiving. In this report, I present a Python script developed for web scraping purposes.

The data is scrapped from the website of the collections on display in Chhatrapati Shivaji Maharaj Vastu Sangrahalaya.

(https://csmvs.in/all-collections/collection-indian-decorative-art-or-sculptures-or-textiles-and-costumes-of-india/objecttype-architectural-fragment/)

Chhatrapati Shivaji Maharaj Vastu Sangrahalaya, formerly known as the Prince of Wales Museum of Western India, is a prominent museum situated in Mumbai, Maharashtra, India. Renamed in honor of the legendary Maratha warrior king, Chhatrapati Shivaji Maharaj, the museum stands as a beacon of cultural heritage and historical significance in the heart of Mumbai.

Established in the early 20th century, the museum boasts an extensive collection of artifacts, artworks, and archaeological treasures spanning various periods and civilizations. Its diverse exhibits encompass a wide range of subjects, including art, sculpture, archaeology, natural history, and decorative arts, offering visitors a comprehensive insight into India's rich cultural tapestry.

Code

The Python script employs the requests library to retrieve HTML content from web pages and the BeautifulSoup library for HTML parsing and content extraction. The script consists of two main functions:

scrape_website(url): This function is designed to scrape content from a given URL. It first sends a GET request to the URL and checks for a successful response. Upon success, it uses BeautifulSoup to parse the HTML content and searches for a specific division identified by its ID and class attributes. If the division is found, the function extracts all links within it and iterates through each link to scrape content using the scrape_page_content(url) function. The extracted content is then concatenated and returned.

```python
import requests
from bs4 import BeautifulSoup
import re

def scrape_website(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.content, 'html.parser')

            # Find the division with the specified id and class
            division = soup.find('div', {'id': 'col-157638099', 'class': 'col collectionDataHolder medium-9 small-12 large-10'})
            if division:
                # Find all links within the division
                links = division.find_all('a', href=True)

                # Extract URLs and scrape content from each link
                content = []

                for link in links:
                    absolute_url = url + link['href'] if link['href'].startswith('/') else link['href']
                    link_content = scrape_page_content(absolute_url)
                    if link_content:
                        content.append(link_content)
                return "\n\n".join(content)
            else:
                print("Division not found.")
                return None
        else:
            print("Failed to fetch the landing page.")
            return None

    except Exception as e:
        print(f"An error occurred: {e}")
        return None
```

scrape_page_content(url): This function is responsible for scraping content from a specific page URL. It sends a GET request to the URL, parses the HTML content using BeautifulSoup, and searches for a division with a particular class attribute. If the division is found, it retrieves the text content from it and returns it.

```python
def scrape_page_content(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.content, 'html.parser')

            # Find the division with the specified class
            division = soup.find('div', {'class': 'large-6 col'})
            if division:
                # Extract text content
                content = division.get_text(' | ', strip=True)
                return content
            else:
                print("Division not found.")
                return None
        else:
            print(f"Failed to fetch the page at {url}")
            return None
    except Exception as e:
        print(f"An error occurred: {e}")
        return None
```

Text Vector Database Creation

The creation of a FAISS vector database serves enables fast and effective document retrieval based on semantic similarity.

The Python script leverages several modules from the langchain_community and langchain packages to facilitate the creation of a vector database. Here's a breakdown of the key components:

Document Loading: The script begins by loading textual content from a file named "museum_content.txt". This file contains information relevant to museum exhibits or artifacts.

Text Processing: To handle potentially large documents efficiently, the loaded content undergoes a process of text splitting. This is accomplished using a RecursiveCharacterTextSplitter, which segments the documents into smaller, manageable chunks.

Language Embeddings: The script utilizes the HuggingFaceEmbeddings module to generate embeddings for the text. This module employs a pre-trained transformer-based sentence-transformer model from the HuggingFace transformers library to encode textual information into dense vector representations.

Vector Database Creation: The core functionality of the script involves the creation of a vector database using FAISS (Facebook AI Similarity Search). FAISS is a powerful library for efficient similarity search and clustering of high-dimensional vectors. The script constructs the vector database (db) from the processed texts and their corresponding embeddings.

Database Persistence: Upon creation, the vector database is saved locally using the specified path (DB_FAISS_PATH) and is named index.faiss automatically. This ensures that the database can be easily accessed and utilized in subsequent tasks without the need for repeated computation.

Development of a Question-Answering Chatbot

To achieve the project objective, we utilized a combination of Python libraries and frameworks, including LangChain and Chainlit, which are specialized for developing applications powered by language models.

The following key components were implemented:

Custom Prompt Template:A custom prompt template was defined to guide the chatbot in formulating responses to user queries. The template includes placeholders for context and question, ensuring consistency and clarity in the bot's responses.

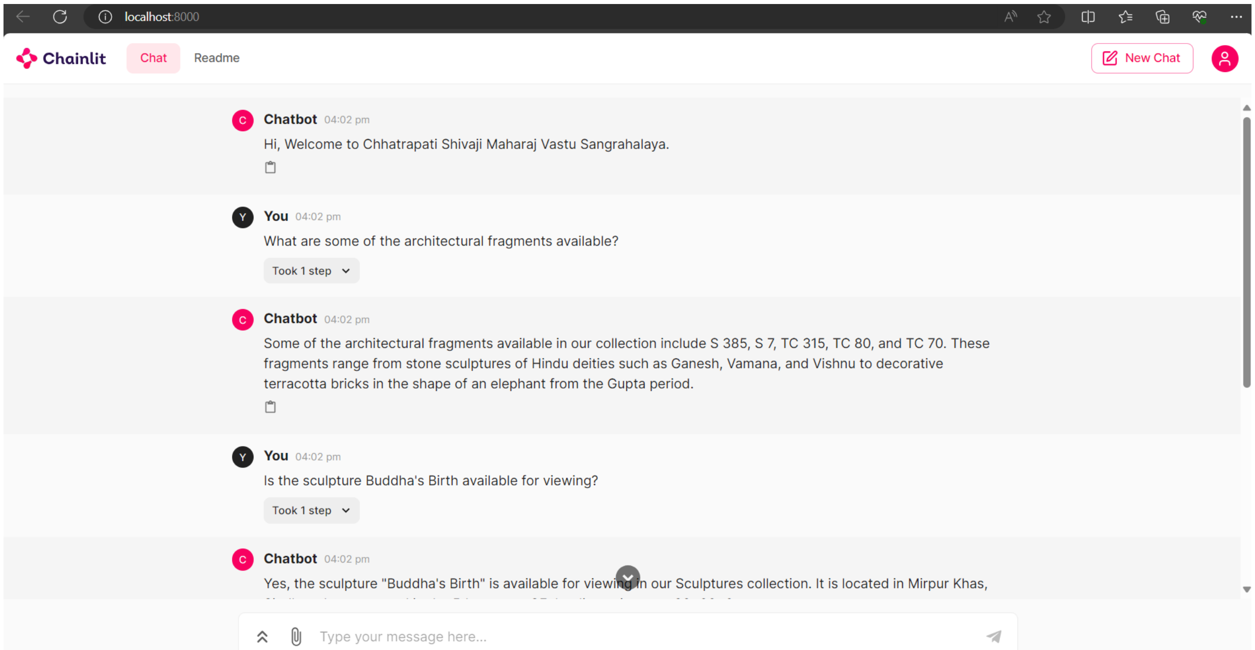
Retrieval QA Chain: The retrieval-based QA chain was configured using LangChain's RetrievalQA class. This chain integrates a language model (LLM), retriever, and prompt template to retrieve relevant information from a database in response to user queries. The top five similarity search results are considered as context.

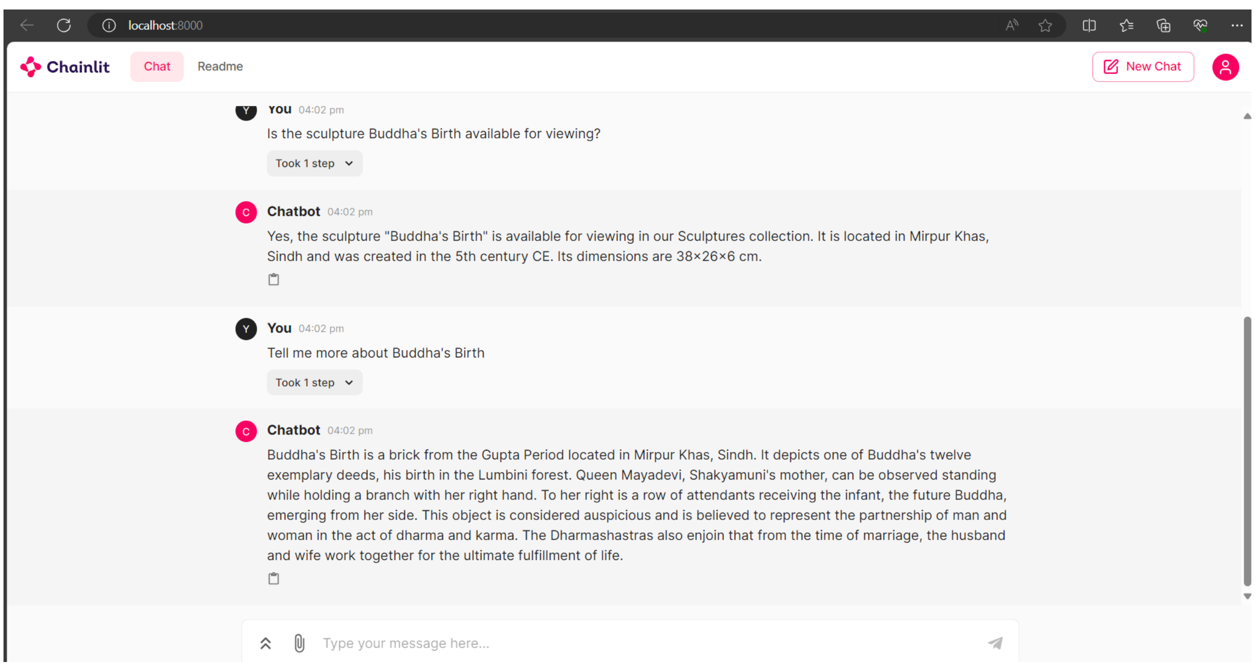
Language Model (LLM) Initialization: The GPT-3 language model from OpenAI is loaded using the API key loaded from the YAML file. It serves as the primary model for generating responses to user queries.

QA Bot Setup: The QA bot setup involved loading embeddings, setting up the FAISS vector store (database), and configuring the retrieval QA chain. This bot is responsible for processing user queries and providing accurate responses based on the retrieved information.

Message Handling: The chatbot's message handling functionality was implemented using Chainlit's asynchronous message handling capabilities. This functionality allows the bot to process user input, retrieve relevant information using the QA chain, and display the response (along with any source documents found during the retrieval process - optional)

Output


localhost:8000
Chainlit
Chat
Readme
New Chat
Chatbot 04:02 pm
Hi, Welcome to Chhatrapati Shivaji Maharaj Vastu Sangrahalaya.
You 04:02 pm
What are some of the architectural fragments available?
Took 1 step
Chatbot 04:02 pm
Some of the architectural fragments available in our collection include S 385, S 7, TC 315, TC 80, and TC 70. These fragments range from stone sculptures of Hindu deities such as Ganesh, Vamana, and Vishnu to decorative terracotta bricks in the shape of an elephant from the Gupta period.
You 04:02 pm
Is the sculpture Buddha's Birth available for viewing?
Took 1 step
Chatbot 04:02 pm
Yes, the sculpture "Buddha's Birth" is available for viewing in our Sculptures collection. It is located in Mirpur Khas,
Type your message here...


localhost:8000
Chainlit
Chat
Readme
New Chat
You 04:02 pm
Is the sculpture Buddha's Birth available for viewing?
Took 1 step
Chatbot 04:02 pm
Yes, the sculpture "Buddha's Birth" is available for viewing in our Sculptures collection. It is located in Mirpur Khas, Sindh and was created in the 5th century CE. Its dimensions are 38×26×6 cm.
You 04:02 pm
Tell me more about Buddha's Birth
Took 1 step
Chatbot 04:02 pm
Buddha's Birth is a brick from the Gupta Period located in Mirpur Khas, Sindh. It depicts one of Buddha's twelve exemplary deeds, his birth in the Lumbini forest. Queen Mayadevi, Shakyamuni's mother, can be observed standing while holding a branch with her right hand. To her right is a row of attendants receiving the infant, the future Buddha, emerging from her side. This object is considered auspicious and is believed to represent the partnership of man and woman in the act of dharma and karma. The Dharmashastras also enjoin that from the time of marriage, the husband and wife work together for the ultimate fulfillment of life.
Type your message here...