Report on URL Based Query System

Project Overview

The URL Based query system is designed to allow users to extract information from a specified web page and ask questions about its content. The system fetches the text from the URL provided by the user, processes it to generate embeddings, and utilizes a language model to generate answers based on user queries. This application integrates natural language processing (NLP) and vector database technologies to facilitate effective information retrieval.

Objectives

- To extract text content from a specified URL and its linked pages.
- To create embeddings of the extracted content for efficient search and retrieval.
- To answer user queries based on the content of the provided URL.

Models Used

1. Sentence Transformers

Model: sentence-transformers/all-MiniLM-L6-v2

This model is used for generating high-quality embeddings of textual content. It provides vector representations of the text that can capture semantic similarity, allowing for effective querying and retrieval.

The embeddings are generated by passing the text chunks extracted from the URL to this model. The model processes the text and outputs a numerical vector representing its meaning.

2. Hugging Face Text Generation Model

Model: distilbart-cnn-12-6

This model is employed to generate coherent and contextually relevant answers based on the user's queries and the retrieved content from the URL.Once a query is processed, the system retrieves the most relevant text chunk(s) based on their embeddings and passes this information to the text generation model, which constructs a detailed response.

Implementation Steps

Step 1: Setting Up the Environment

- **Frameworks**: FastAPI for building the backend API and Streamlit for creating a user-friendly frontend interface.

- **Dependencies**: Required libraries include FastAPI, Uvicorn, BeautifulSoup for web scraping, Requests for HTTP requests, Transformers from Hugging Face for model handling, ChromaDB for storing embeddings, and Streamlit for the web interface.

Step 2: Extracting Text from the URL

- A function is implemented to fetch content from the provided URL using the Requests library. The BeautifulSoup library is used to parse the HTML content and extract visible text. The function also processes links to gather text from connected pages.

Step 3: Chunking the Text

- The extracted text is divided into smaller, manageable chunks using a specified chunk size. This ensures that the model can handle the text effectively during embedding generation.

Step 4: Generating Embeddings

- For each text chunk, the Sentence Transformers model is invoked to produce vector embeddings. These embeddings are stored in a ChromaDB vector database for efficient querying.

Step 5: Processing User Queries

- The system listens for POST requests on the /query endpoint, where users can submit their questions.
- The query is converted into an embedding using the same model, and the system searches the vector database to find the most relevant text chunks.

Step 6: Generating Answers

- The retrieved text chunks are fed into the Hugging Face text generation model to produce a coherent answer.
- The generated response is returned to the user through the API.

Step 7: Streamlit Interface

- A simple web interface is created using Streamlit, where users can input a URL and a query.Upon submission, the app communicates with the FastAPI backend to process the request and display the results.