Overview

ChatBot is a sophisticated chat application powered by a state-of-the-art Language Model (LLM). It leverages advanced techniques for information retrieval and augmentation to provide users with a seamless and efficient chatting experience. The application's database is structured as a vector database, specifically using Chroma DB, and document embeddings are generated using LangChain and OpenAI's "text_ada_002."

Installation

To use ChatBot, you need to have Python installed on your system. Additionally, you can install the required dependencies by running:

pip install -r requirements.txt

Ensure that you have the necessary versions of libraries as specified in the requirements.txt file.

Usage

1. Running the Application

Start the ChatBot application using the following command:

python ChatBot.py

This will launch the chat interface, allowing users to interact with the system.

2. User Query Handling

- Users can input queries into the chat interface.

- ChatBot employs a retrieval augmentation technique to search through the vector database (Chroma DB) and provide relevant information.

3. Database Structure

- The database is organized in the form of a vector database using Chroma DB.

- Document embeddings are generated using LangChain and OpenAI's "text_ada_002" to represent text documents as vectors.

Features

1. Retrieval Augmentation

- ChatBot uses advanced retrieval augmentation techniques to enhance the accuracy and relevance of search results.

2. Vector Database (Chroma DB)

- The database is structured as a vector database, allowing for efficient storage and retrieval of information.

3. Document Embeddings

- Document embeddings are created using LangChain and OpenAI's "text_ada_002" to represent text documents as vectors.

4. User-Friendly Chat Interface

- The chat interface is designed to be user-friendly, enabling easy interaction with the application.

Dependencies

ChatBot relies on the following Python libraries and versions:

Langchain == 0.0.194

LangChain is a comprehensive framework designed for the development of applications powered by language models. It facilitates the creation of context-aware applications that can connect language models to various sources of context, such as prompt instructions, few-shot examples, or relevant content, enabling them to provide more accurate and relevant responses. Additionally, LangChain enables applications to reason by relying on language models to determine how to respond based on the provided context and what actions to take. The framework comprises several components, including LangChain Libraries, which consist of Python and JavaScript libraries containing interfaces, integrations for various components, a basic runtime for combining these components into chains and agents, and pre-built implementations of chains and agents. Furthermore, LangChain offers LangChain Templates, a collection of easily deployable reference architectures for a wide range of tasks, LangServe, a library for deploying LangChain chains as a REST API, and LangSmith, a developer platform that facilitates debugging, testing, evaluation, and monitoring of chains built on any LLM framework while seamlessly integrating with LangChain.

For our ChatBot application we utilize LangChain **framework. We use Langchain version** 0.0.194 for our application. **It provides the us platform to adapt language model with flexibility to our specific context. We utilize some of the core components within the langchain to build context-aware language models systems. To achieve this, we use core components including LLM interface, chains, prompt templates, retrieval modules, memory. In case of LLM interface, we interface with proprietary models such as GPT to make API calls. We leverage gpt-3.5-turbo as our base generator model.** *Chains* are the fundamental principle that holds various AI components in LangChain to provide context-aware responses. A chain is a series of automated actions from the user's query to the model's output. Chain in our case implements Program-Aided Language Models (PAL) for generating code solutions. This is utilized to answer math or code related questions in our interface. PALChain reads complex math problems (described in natural language) and generates programs (for solving the math problem) as the intermediate reasoning steps, but offloads the solution step to a runtime such as a Python interpreter.

Streamlit == 0.63.0

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Streamlit allows you to create a stunning-looking application with only a few lines of code.

In our chatbot application we utilize streamlit version 0.63.0. To do so we first install streamlit using pip command. The syntax for installation is pip install streamlit. We then create logic for our chatbot. his script will handle user inputs, process them, and generate appropriate responses.

Docker == 3.1

Docker Desktop is a one-click-install application for your Mac, Linux, or Windows environment that lets you build, share, and run containerized applications and microservices.

We use Docker version 3.1 for ChatBot application. It is hosted in Linux environment (ubuntu22.04). The services we run on docker environment is streamlit. The application runs on port 8501. The resources for deployment in terms are as follows. For drivers we use NVidia, and count is set for all, and capabilities is set to GPU. In relation to nginx, the image is nginx and volumes are - ./nginx/streamlit.conf:/etc/nginx/conf.d/default.conf- /etc/letsencrypt/live/patentlab.cse.msstate.edu/privkey.pem:/etc/letsencrypt/live/patentlab.cse.msstate.edu/privkey.pem- /etc/letsencrypt/live/patentlab.cse.msstate.edu/cert.pem:/etc/letsencrypt/live/patentlab.cse.msstate.edu/cert.pem. The port for nginx is 4000:443

Gitpython == 3.1.40

GitPython is a python library used to interact with Git repositories. GitPython provides object model read and write access to your git repository. Access repository information conveniently, alter the index directly, handle remotes, or go down to low-level object database access with big-files support.

For our chatbot application, GitPython version 3.1.40 is used in various ways, depending on our specific requirements. We particularly use it for version control for chatbot code which include tasks such as cloning a repository containing chatbot code, pulling updates from remote repository, committing changes made to the codebase, and pushing those changes back to the remote repository.

JobLib == 1.0.0

Joblib offers a suite of tools aimed at facilitating lightweight pipelining in Python. Its primary features include transparent disk-caching of functions, enabling lazy re-evaluation through the memoize pattern, and providing straightforward support for parallel computing. It's designed with optimization for speed and robustness, especially when handling large datasets, with specific enhancements tailored for NumPy arrays. This makes Joblib particularly useful for efficiently caching function outputs, performing parallel computations, and managing resources in Python applications, particularly those involving data-intensive tasks.

In our chatbot application we use joblib version 1.0.0. We use it for model caching, for example, we cache the results of user inputs that utilizes the LLM model to avoid re-running it for every user query. Similarly, we also use Joblib for resource management task such as large dataset or pretrained models. By efficiently serializing and deserializing these resources, Joblib can reduce memory usage and improve performance.

markdown-it-py == 2.1.0

Markdown-it is a robust, extensible, fast, and easy-to-use markdown parser for Node.js and the browser. This library follows the CommonMark specification and provides a variety of syntax extensions and functionality for processing markdown text. With Markdown-it, you can generate HTML or render tokens sequence from your markdown text. It also allows you to replace the existing syntax rules, add new ones, and includes a variety of plugins for extending the library's functionality.

For our Chatbot application we use Markdown-it version 2.1.0 to generate HTML. We do this via API call. The package we use is markdown-it-py package. The transformation take place by first converting raw text into tokens and then converting to other formats using renderers.

toml = = 2.14.0

TOML, short for "Tom's Obvious, Minimal Language," is a human-readable data serialization format. It aims to be easy to read and write due to its straightforward syntax, while also being unambiguous for parsing. TOML is commonly used for configuration files, particularly in projects written in programming languages such as Rust, Python, and Go, among others.TOML syntax is based on key-value pairs, with support for various data types like strings, integers, floating-point numbers, arrays, and nested tables. It organizes data in a hierarchical structure, making it suitable for representing complex configurations.

In our chatbot application, TOML version 2.14.0 is primarily used for configuration purposes. It helps organize settings such as API keys, language resources, intent mappings, and plugin configurations. TOML's hierarchical structure enables clear organization of settings, facilitating easier maintenance and customization of the chatbot's behavior. For example, a sample configuration for our Chatbot application using TOML is as follows: api_key = "YOUR_API_KEY" language = "en" max_responses = 5

tqdm == 4.4.1

TQDM stands for "TQDM Quick Documentation Markup." It is a Python library that provides a fast, extensible progress bar for loops and iterables. TQDM stands for "Taichi QUick Documentation Markup." TQDM provides a simple way to add progress bars to your loops in Python, allowing you to track the progress of your code's execution. It's particularly useful when working with long-running tasks or iterating over large datasets, as it provides real-time feedback on the progress of your code.

Our chatbot application involves heavy data processing tasks such as data loading, preprocessing, or model training, for these task we use TQDM version 4.4.1 to display progress bars for these tasks. This provides real-time feedback to users on the progress of the processing.

NLTK == 3.6.2

NLTK stands for Natural Language Toolkit, which is a comprehensive library in Python used for natural language processing (NLP) tasks. It provides tools and resources for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, parsing, and semantic reasoning.

In our chatbot application, NLTK is utilized for various purposes such as Text Processing and Language Understanding. We particularly utilize NLTK version 3.6.2. For text processing task, NLTK version 3.6.2 preprocess and clean the text input from users, including tokenization (breaking text into words or sentences), stemming (reducing words to their root forms), and lemmatization (reducing words to their base or dictionary forms). While for language understanding task, NLTK version 3.6.2 is used to analyze the user's input to understand

the intent and extract relevant information. This includes part-of-speech tagging to identify the grammatical structure of sentences, named entity recognition to identify entities like names, dates, and locations, and sentiment analysis to determine the sentiment expressed in the text.

Pandas == 1.4.3

Pandas is a Python library primarily used for data manipulation and analysis. It provides data structures and functions to efficiently handle structured data, such as tables and time series, making it particularly useful for tasks like data cleaning, transformation, exploration, and visualization.

In the context of our chatbot application, we use Pandas version 1.4.3 for data processing. Pandas is utilized to preprocess and clean datasets before training machine learning models for the chatbot. This includes tasks such as removing duplicates, handling missing values, and transforming data into a format suitable for analysis.

Scipy == 0.9.0

Scipy is a Python library used for scientific computing and technical computing. It builds on top of NumPy, providing additional functionality for tasks such as optimization, integration, interpolation, linear algebra, signal processing, and statistical analysis.

In the context of our chatbot application, Scipy version 0.9.0 is be utilized for **Statistical** analysis. For **Statistical analysis we use** functions for computing descriptive statistics, hypothesis testing, probability distributions, and correlation analysis. We do this to analyze user data collected by the chatbot and derive insights from it. Our chatbot collects user feedback through a rating system, and we want to analyze the distribution of ratings to understand user satisfaction. In such case we use Scipy's statistical functions to calculate descriptive statistics like the mean, median, and standard deviation of the ratings. Additionally, we perform hypothesis tests to compare the ratings between different user groups or time periods. This analysis can help us to identify areas for improvement and make data-driven decisions to enhance user experience.

Numpy == 1.15.0

NumPy is a powerful Python library for numerical computing that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. It is widely used in scientific and technical computing tasks, including data manipulation, linear algebra, statistical analysis, and more.

In the context of our chatbot application, NumPy version 1.15.0 is leveraged for various purposes, such as: Data preprocessing: NumPy arrays can be utilized to represent and manipulate textual or numerical data collected by the chatbot. For example, user inputs or feedback ratings can be stored in NumPy arrays for further processing.

Pillow == 10.0.0

Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.

In the context of a chatbot application, Pillow version 10.0.0 is used for various image processing tasks, such as resizing images for UI interface. We use 3 images. These images are for mascot, logo and banner in the interface.

Requests == 2.17.0

The Requests library is a popular Python library used for making HTTP requests in web applications. It provides a simple and elegant API for sending various types of HTTP requests, such as GET, POST, PUT, DELETE, etc., and handling the responses.

In the context of our chatbot application, we can use the Requests library version 2.17.0 to interact with external APIs, fetch data from remote servers, or perform actions such as sending user inputs to a backend server for processing. For example, we utilize OPENAI API in our chatbot. First, we import **Requests Library** for example **import requests. This allows python to make HTTP requests to web servers. We then define a function that takes in the API key that we then utilize to access the service.** This function encapsulates the logic for querying the OpenAI API. It takes a prompt (input text) as an argument and returns the generated text from the API response.

PyYAML == 5.1

PyYAML is a Python library that enables parsing and emitting YAML (YAML Ain't Markup Language) files. YAML is a human-readable data serialization format that is commonly used for configuration files, data exchange, and other structured data representation tasks.

In a chatbot application, PyYAML version 5.1 is utilized for purposes, such as **Configuration Management** PyYAML is used to parse YAML configuration files that define various settings and parameters for the chatbot application. This includes defining conversation flows, setting up external API credentials, specifying user interface preferences, and more.

PyArrow == 13.0.0

PyArrow is a Python library designed for working with large datasets efficiently, particularly in the context of data serialization and inter process communication. It provides functionalities for converting between different data formats, such as Pandas DataFrames and Apache Parquet files, with high performance and low memory overhead.

In the context of a chatbot application, PyArrow version 13.0.0 is utilized for Integration with Pandas. PyArrow seamlessly integrates with Pandas, a popular data manipulation library in Python. This allows the chatbot application to process large datasets efficiently, perform computations, and generate insights using Pandas DataFrames.