SELF-HOSTED CONTEXT AWARE CHATBOT

Chris Andrei Irag, Hernel Juanico
BS Data Science - DS3A
University of Science and Technology of Southern Philippines
Cagayan de Oro, Philippines
irag.chrisandrei@mailbox.org, juanico.hernel064@gmail.com

ABSTRACT

This concept paper presents the design and implementation of a self-hosted, context-aware chatbot system tailored for academic environments. Leveraging advancements in Large Language Models (LLMs) and open-source frameworks, the proposed system is designed to integrate multiple LLM services, including OpenAI, Gemini, Llama, and other open-sourced models, to create a context aware agent. The chatbot is implemented using Python and deployed via Cloudflare Tunnels, ensuring network security and global accessibility.

***K*eywords** Generative AI · Chat robots · Large Language Models (LLMs) · Self-hosting

1 Introduction

Generative AI represents a significant technological advancement in recent years, enabling computational systems to autonomously generate content derived from diverse data sources. These systems are grounded in advanced machine learning models that identify underlying patterns in data and transform them into various modalities. Notable applications of Generative AI include Large Language Models, which utilize techniques such as vectorization and tokenization to predict and generate coherent textual content.

This implementation is a product of academic coursework conducted as part of the Data Science Elective 1: Generative AI. It explores a cost-efficient framework for deploying generative models by leveraging locally available resources without the requirement of expensive workstations or subscriptions to perform the job. This benefits repurposed hardware, such as outdated laptops and desktop computers, alongside a basic internet connection, offering an alternative to reliance on commercial cloud-based solutions.

2 Implementation

This documentation provides a comprehensive overview of the implementation process, encompassing key steps such as developing the application using Streamlit—a Python framework designed for the rapid deployment of web-based applications—writing and structuring the application's code, integrating large language models (LLMs) for inference, enhancing LLMs with context awareness, identifying appropriate hardware resources, and deploying the application to the cloud via Cloudflare Tunnels.

2.1 Building the application

The web application is written in Python. The source code can be found here.

`https://github.com/iragca/GenAI-portfolio1-2`

Importing Streamlit into the Python script is essential for running the application as a web-based interface and utilizing its functionalities. The application's interface draws inspiration from the chat interface of ChatGPT, providing users with a simple and intuitive experience. It is designed to easily switch between different LLMs, each with its own

schema, referring to the structured approach an LLM uses to process inputs in a consistent manner. Additionally, the application features visualization tools, such as displaying the number of tokens used in the current conversation and other prompt metadata, making it well-suited for benchmarking various LLMs in terms of pricing and performance.

The documentation for Streamlit can be found at:

`https://docs.streamlit.io/develop`

2.2 Integrating LLMs

The LLM models integrated into this app include OpenAI models and Google's Gemini. However, the app is not restricted to supporting only these two models. To enable inference, the application leverages LangChain, a framework for building applications that integrate LLMs with built-in API connectors and advanced functionalities.

Implementing context understanding in LLMs means that there is a need to implement memory, as LLMs are stateless in nature. Current memory solutions imitate that of the human brain, consisting of long-term and short-term memory [1]. This application however is only designed for short-term memory. LangChain provides a Memory module to equip LLMs with memory for context awareness. In this application, a custom implementation was used, relying on Streamlit's session state functionality and Python lists to achieve similar context-aware behavior.

The documentation for LangChain and its Memory module may be found at:

`https://python.langchain.com/v0.1/docs/modules/memory/`

2.3 Hardware and Deployment

The hardware requirements for running this app are not clearly defined, as there are no extensive benchmarking being done beyond testing it on a single desktop machine. However, based on observations, the app operates with an average CPU usage of less than 3% using a desktop powered by an Intel i5-8400 processor. This estimate also accounts for other self-hosted services running simultaneously on the same machine. The compute power demand primarily arises from running the Python application. The challenge of LLM inference has been addressed by leveraging cloud inference services, such as from OpenAI servers. As a result, this application can efficiently run on a low-end computer or a machine capable of web browsing.

Once the Streamlit app is running locally on the machine, it provides a convenient local network endpoint for access. To make the application accessible on the World Wide Web, Cloudflare Tunnels offers a solution by securely exposing the local endpoint without requiring the user to open ports on their network. It is noted that that the user shall require to own an internet domain to use with Cloudflare Tunnels. Cloudflare provides domain registration services at competitive pricing. For instance, the domain utilized for this web application was acquired for $4.16 USD per year at the time of purchase.

The documentation for Cloudflare Tunnels can be found at:

`https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/`

3 Discussion

Other LLMs, including open-source models, can also be integrated into the application. These models can be hosted either on the user's own machine or through cloud services like Replicate. Additionally, running the application within a container, such as Docker, is feasible as this approach offers benefits, including enhanced security, modularization, and compartmentalization. Since this application is deployed using retail hardware, this implementation lacks the scalability provided by cloud services. As a result, the application is best suited for personal use or a small group of users including small businesses or for academic research.

Future development of the application could incorporate long-term memory capabilities, Reasoning and Action (ReAct) [2], and Retrieval-Augmented Generation (RAG) [3]. These advancements would enhance its functionality, transforming it into a robust personal assistant. Instead of relying on costly subscriptions, the application could adopt a pay-as-you-use model, providing accessible and affordable multimodal AI tools. Additionally, if a self-hosted local LLM model is used,

the user retains full control over the model, including parameters such as temperature, context limits, and other settings. This also allows for greater flexibility, avoiding the restrictive policies often enforced by cloud services.

4 Conclusion

This paper presents the development of a web-based application implemented in Python, leveraging large language models (LLMs) with context awareness and short-term memory capabilities. It details the hardware requirements and the configuration process for deploying the application using personal hardware and Cloudflare for web access. Furthermore, potential advancements and additional functionalities to enhance the application's capabilities are explored.

References

[1] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, March 2023. arXiv:2210.03629.

[3] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions, June 2023. arXiv:2212.10509.