Conversational User Interface and LLMs

Vasishtha Pandya

July 20, 2024

1 Abstract

In contemporary digital environments, traditional user interfaces often fall short in providing intuitive and natural interactions, leading to user frustration and decreased engagement. Conversational User Interfaces (CUIs) powered by artificial intelligence (AI) have emerged as a promising solution, enabling users to interact with digital systems using natural language. However, these interfaces face challenges such as understanding user intent, generating accurate responses, and maintaining coherent dialogues. This project aims to address these challenges by designing and implementing AI-driven CUIs that offer seamless and engaging interactions. The project employs advanced natural language processing techniques, Retrieval-Augmented Generation (RAG), and Large Language Model (LLM) APIs to enhance conversational AI capabilities.

2 Introduction

Traditional user interfaces (UIs) often fail to provide intuitive interactions, leading to user frustration. With the rise of artificial intelligence (AI), conversational user interfaces (CUIs) have emerged as a solution, allowing users to interact using natural language. However, despite significant advancements, existing CUIs still encounter several critical challenges:

- **Understanding User Intent:** Current CUIs struggle with ambiguous or context-dependent language, making it difficult to accurately interpret user queries. [3]
- **Accurate Response Generation:** Generating responses that are both accurate and contextually appropriate remains a significant hurdle. [4]
- **Maintaining Coherent Dialogues:** Ensuring coherence over multiple interactions and maintaining the context throughout a conversation is a complex task. [5]

This project aims to address these challenges by leveraging state-of-the-art techniques in natural language processing (NLP) and AI. The primary

objective is to design and implement AI-driven CUIs that enable seamless and engaging user interactions. By integrating advanced methods such as Retrieval-Augmented Generation (RAG) and the LangChain framework, we aim to achieve the following specific goals:

- Enhancing dialogue understanding with advanced NLP techniques to better interpret user intent.
- Improving response generation using RAG and the LangChain framework to produce accurate and contextually appropriate responses.
- Developing robust dialogue management systems within the LangChain framework to maintain coherent and context-aware conversations.

This research contributes to the field of conversational AI by proposing innovative solutions to overcome existing limitations, thus paving the way for more effective and user-friendly CUIs.

3 Methodology

The methodology focuses on enhancing user intent understanding, accurate response generation, and coherent dialogue maintenance through advanced techniques and tools. Additionally, Streamlit is utilized as the user interface for creating an interactive and user-friendly application.

3.1 Data Collection

The first step in the methodology involves data collection. Data is collected from various sources relevant to the specific domain of the CUI, satisfying diverse user queries. This data is then converted to embeddings and stored in a vector database such as FAISS to enhance the performance of Retrieval-Augmented Generation (RAG).

3.2 Model Development and Integration

This can be divided into three subsections:

3.2.1 Understanding User Intent

Understanding user intent is crucial for effective conversational AI. Current CUIs often struggle with ambiguous or context-dependent language, leading to misinterpretations.

- **Natural Language Processing (NLP) Techniques**:
  - **Tokenization and Lemmatization**: Breaking down sentences into individual words or phrases and reducing them to their base forms.

  - **Named Entity Recognition (NER)**: Identifying and classifying key entities in the text (e.g., names, dates, locations).
  - **Part-of-Speech (POS) Tagging**: Assigning parts of speech to each word to understand its grammatical structure.

- **GPT-2 Model**: Fine-tune the GPT-2 model on domain-specific data to improve its ability to understand and predict user intent accurately.

3.2.2 Accurate Response Generation

Generating accurate and contextually appropriate responses is a significant challenge for conversational AI.

- **Retrieval-Augmented Generation (RAG)**:
  - **Integration**: Implement RAG to enhance information retrieval from large datasets. RAG combines the strengths of retrieval-based and generative models, ensuring responses are grounded in accurate and relevant information.

- **LLM API Utilization**:
  - **State-of-the-Art LLMs**: Employ APIs like Llama3 for response generation. These models have been pretrained on vast datasets, enabling them to generate more accurate and contextually relevant responses.
  - **API Fine-Tuning**: Fine-tune LLM APIs using domain-specific data to improve their accuracy and contextual relevance. This process involves training the models on data specific to the domain to enhance their performance.

3.2.3 Maintaining Coherent Dialogues

Ensuring conversations remain coherent over multiple interactions is challenging, particularly in complex or multi-turn dialogues.

- **Dialogue Management Systems**:
  - **LangChain Framework**: Develop robust dialogue management systems using the LangChain framework. LangChain provides tools and libraries for managing dialogue state and context, ensuring coherence over extended interactions. [6]
  - **State Tracking**: LangChain maintains a robust memory of the dialogue state throughout interactions. This memory stores crucial information such as user preferences, context from previous turns, and ongoing tasks. It allows the system to track the progression of conversations and maintain coherence.

4 Technical Details of Prototype Design

The prototype for testing the model is designed using Streamlit as the user interface. The application integrates Langchain and RAG to facilitate seamless user interactions. The prototype's architecture includes:

- **Frontend:** Built with Streamlit, providing an interactive interface for users to input queries and receive responses.
- **Backend:** Incorporates Langchain for dialogue management and RAG for response generation. Embeddings are stored in a FAISS vector database for efficient retrieval.
- **LLM Integration:** Utilizes models like Llama3 and GPT-2 for generating responses. The prototype allows for comparative analysis between these models.

The implementation involved the following libraries:

- **Streamlit:** Used for building the user interface.
- **PyPDF2:** Utilized for reading PDF documents.
- **Huggingface Transformers:** Specifically, GPT-2 for language model operations.
- **Sentence Transformers:** Used for converting sentences to embeddings.
- **FAISS:** Employed as the vector database for efficient retrieval.
- **Langchain:** Utilized for dialogue management and integration with RAG.
- **Ollama Model:** Included for utilising llama3 capabilities, providing enhanced generative and retrieval functions.
- **OllamaEmbeddings:** Utilized for generating and managing embeddings for llama3 model, improving the retrieval process in the backend.

5 Workflow of proposed solution

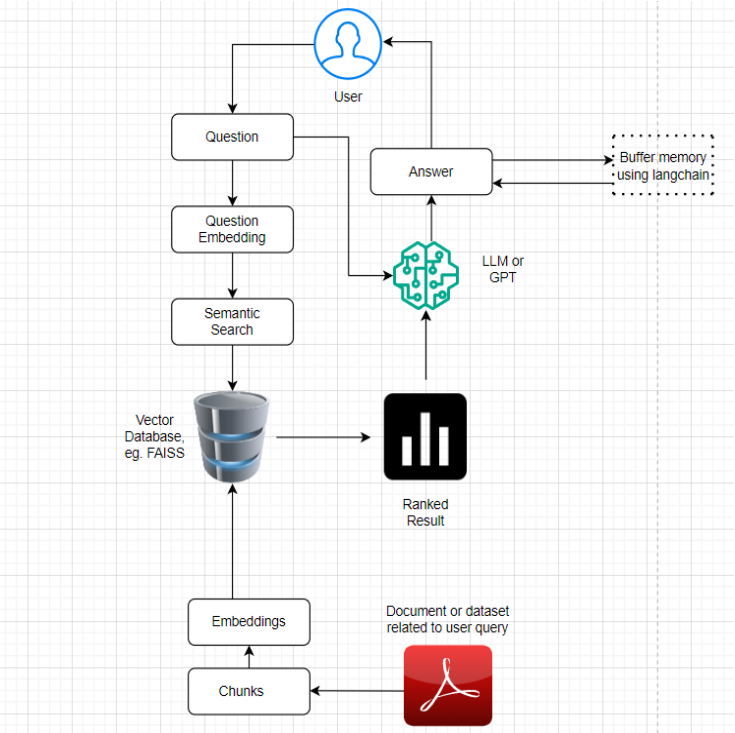

Figure 1: RAG using langchain

The Retrieval-Augmented Generation (RAG) approach leverages both retrieval-based and generative models to produce contextually relevant and accurate responses. The workflow of RAG using LangChain involves several key steps, which can be categorized into data processing, model development, and response generation.

5.1 Data Collection and Embedding Storage

- **Data Collection**: Collect data from various sources relevant to the specific domain. This data includes documents, articles, user queries, and other textual information that can be useful for generating responses.
- **Embedding Storage**: Convert the collected textual data into vector embeddings using models such as BERT or Sentence Transformers. These embeddings are stored in a vector database like FAISS, enabling efficient retrieval of relevant information based on user queries. [2]

5.2 Query Processing

- **User Query Input**: The user inputs a query through the Streamlit interface.
- **Query Embedding**: The input query is converted into a vector embedding using the same embedding model used for the data. This ensures consistency and relevance during the retrieval process.

5.3 Retrieval Phase

- **Vector Search**: The query embedding is used to perform a vector search in the FAISS database. The most relevant documents or passages are retrieved based on their proximity to the query embedding in the vector space.
- **Context Extraction**: Extract the top-k relevant contexts from the retrieved documents. These contexts will provide the necessary information for generating accurate and relevant responses. [2]

5.4 Generative Phase

- **Input to Generative Model**: The retrieved contexts are concatenated with the user query and fed into a generative model, such as GPT-2 or Llama3. The generative model uses the provided context to produce a response that is both relevant and coherent.
- **Response Generation**: The generative model generates a response based on the input query and the retrieved context. This response leverages the information from the relevant documents to ensure accuracy and contextual relevance.

5.5 Post-Processing

- **Coherence Check**: The generated response is checked for coherence and relevance. LangChain's dialogue management system ensures that the response aligns with the ongoing conversation and maintains coherence over multiple interactions.
- **User Feedback Loop**: The response is presented to the user through the Streamlit interface. User feedback can be collected to further fine-tune the models and improve future responses. [1]

5.6 Dialogue Management

- **State Tracking**: LangChain maintains a memory of the dialogue state throughout interactions. This memory stores crucial information such as user preferences, context from previous turns, and ongoing tasks.

- **Contextual Updates**: Update the dialogue state with new information from the current interaction, ensuring that the system remains context-aware and can handle complex, multi-turn dialogues effectively. [1]

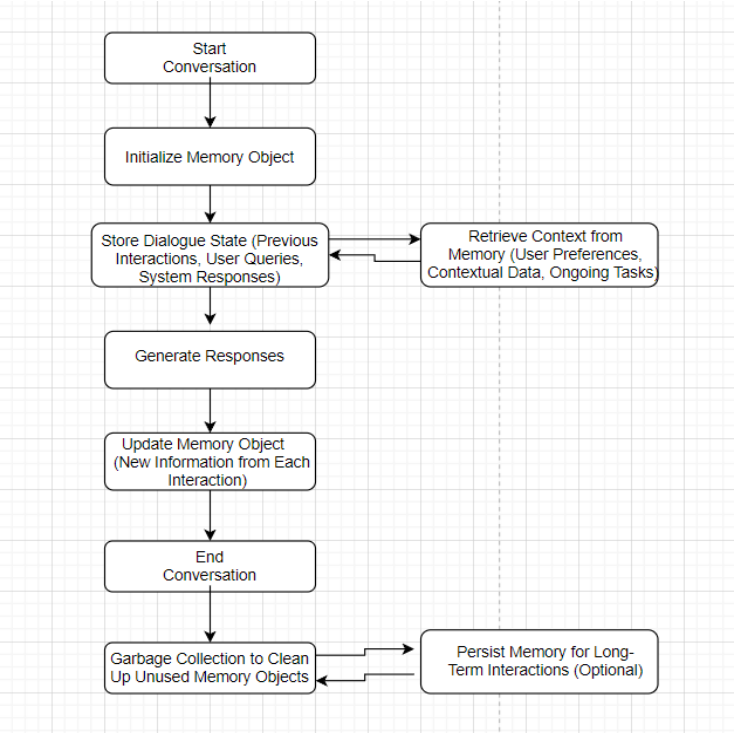

Figure 2: How memory management works

6 Important Formulas

In this section, we discuss key formulas and concepts relevant to the GPT-2 and Llama3 models, which are utilized for response generation in the Retrieval-Augmented Generation (RAG) approach.

6.1 Transformer Architecture

Both GPT-2 and Llama3 are based on the Transformer architecture, which relies on self-attention mechanisms to process and generate language. The core formula for the self-attention mechanism is:

Self-Attention:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where:

- Q is the query matrix.
- K is the key matrix.
- V is the value matrix.
- d_k is the dimension of the keys (scaling factor).

6.2 Language Modeling

GPT-2 and Llama3 are trained to predict the next word in a sequence, given the previous words. This is done using the following formula for language modeling:

Next Word Prediction:

$$P(w_t|w_{1:t-1}) = softmax(W \cdot h_t)$$

Where:

- w_t is the word at position t.
- $w_{1:t-1}$ are the preceding words.
- W is the weight matrix.
- h_t is the hidden state at position t, computed using the Transformer architecture.

6.3 Fine-Tuning for Contextual Relevance

To fine-tune GPT-2 and Llama3 for specific domains, we use a domain-specific dataset. The fine-tuning process adjusts the model's weights to improve performance on the target domain. The loss function typically used for fine-tuning is the cross-entropy loss:

Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{j=1}^{V} y_{ij} \log(\hat{y}_{ij})$$

Where:

- N is the number of examples.
- V is the vocabulary size.
- y_{ij} is the true distribution (one-hot encoded vector).
- $\hat{y}_{ij}$ is the predicted probability distribution.

6.4 Vector Embeddings for Retrieval

In the RAG framework, textual data is converted into vector embeddings for efficient retrieval. The embeddings are generated using models like BERT or Sentence Transformers, and stored in a vector database like FAISS. The similarity between query and document embeddings is calculated using cosine similarity:

Cosine Similarity:

$$cosine\_similarity(a, b) = \frac{a \cdot b}{\|a\|\|b\|}$$

Where:

- a and b are the vector embeddings of the query and the document, respectively.
- $\|a\|$ and $\|b\|$ are the magnitudes of the vectors.

6.5 Response Generation with RAG

In RAG, retrieved documents provide context for the generative model. The final response is generated by conditioning the language model on the query and the retrieved contexts. This involves concatenating the query with the top-k retrieved documents and passing them through the generative model to produce a coherent response:

Contextual Input:

$$Input = [Query; Context_1; Context_2; \ldots; Context_k]$$

Where:

- $Query$ is the user input.
- $Context_i$ are the top-k retrieved contexts.

By integrating these formulas and mechanisms, GPT-2 and Llama3 can generate responses that are contextually relevant, accurate, and coherent, leveraging the strengths of both retrieval and generative approaches.

7 Methodological Differences: Transformers vs. LLMs (LLaMA)

7.1 User Intent Understanding

- **GPT-2:** GPT-2 is a powerful generative model that relies on extensive training data to predict the next word in a sequence. It captures a wide range of contexts and nuances in language, enabling it to understand user intent to a considerable degree. However, it often struggles with ambiguous or highly context-dependent queries due to its relatively fixed and non-specialized nature.

- **LLaMA:** LLaMA, as a newer and more advanced LLM, incorporates several improvements over GPT-2. It utilizes more sophisticated attention mechanisms and larger training datasets, which enhance its ability to understand user intent, particularly in complex or ambiguous queries. Additionally, LLaMA benefits from more refined fine-tuning processes, allowing it to adapt better to specific domains and user intents.

7.2 Accurate Response Generation

- **GPT-2:** GPT-2 generates responses based on the patterns and information it has seen during training. While it can produce accurate and relevant responses, its ability to do so diminishes when dealing with domain-specific queries or when requiring the integration of external, updated information. The accuracy of its responses is limited by the static nature of its training data.

- **LLaMA:** LLaMA improves upon GPT-2 by integrating a more dynamic approach to response generation. It is better equipped to handle domain-specific information and can leverage more extensive and recent datasets. Its enhanced architecture allows for more precise and contextually relevant responses, making it more reliable for specialized applications. LLaMA's ability to access and utilize updated information ensures that its responses are both accurate and current.

7.3 Coherent Dialogue Flow

- **GPT-2:** Maintaining coherent dialogue flow over multiple interactions is challenging for GPT-2. While it can generate individual responses that are coherent and contextually appropriate, it often struggles with tracking long-term context and maintaining consistency throughout extended dialogues. This limitation is due to its lack of explicit memory mechanisms and reliance on short-term context.

- **LLaMA:** LLaMA addresses the issue of dialogue coherence through advanced memory and state-tracking mechanisms. Its architecture allows for better handling of long-term dependencies and context retention across multiple turns in a conversation. By maintaining a robust dialogue state, LLaMA ensures that interactions remain coherent and contextually relevant, even in complex or multi-turn dialogues.

8 Results

In this section, we compare the performance of GPT-2 and LLaMA (LLMs) using BLEU metrics. BLEU (Bilingual Evaluation Understudy) is a metric commonly used to evaluate the quality of machine-generated text by comparing it to one or more reference translations or human-generated texts. It measures

the similarity between the generated text and the reference texts based on n-gram overlap.

8.1 Experimental Setup

We used ChatGPT-4 as the base model for generating standard responses. Specifically, we selected a user query regarding catalytic converters and identified an article on catalytic converters from the internet, which we used for the Retrieval-Augmented Generation (RAG) process. I have framed 5 queries from the document which I am going to ask to the CUIs of both models. The queries are :

- Tell me about Rajasthani Clay Catalytic Converters for Automobiles
- Which domain is it related to?
- Is there any patent or work published on it?
- Tell me more about the background of this field
- Tell me about catalytic converters

8.2 Evaluation Metrics

A set of queries related to catalytic converters was presented to both the LLaMA3 (LLMs) and GPT-2-based conversational user interfaces (CUIs). The BLEU score was calculated by comparing the responses generated by GPT-2 and LLaMA3 to the standard responses generated by ChatGPT-4. **The responses for the different models are enlisted in 'BLEU result.ipynb' file along with the code to calculate metrics.**

8.3 Results

The evaluation yielded the following BLEU scores:

Model	**BLEU Score**
LLaMA3 (LLMs)	0.317
GPT-2	0.1289

Table 1: BLEU Scores for LLaMA3 (LLMs) and GPT-2

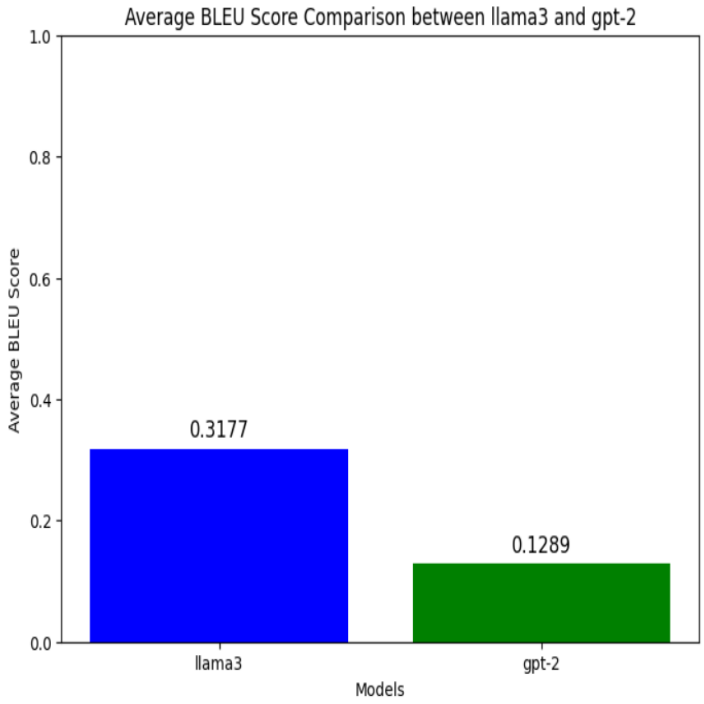

Figure 3: Result showing comparison between GPT-2 and Llama3

These scores indicate the level of similarity between the responses generated by LLaMA and GPT-2 compared to the standard responses from ChatGPT-4. Higher BLEU scores suggest better alignment with the reference responses, reflecting improved performance in generating contextually appropriate and accurate responses, especially in the domain of catalytic converters.

8.4 Discussion

The higher BLEU score achieved by LLaMA indicates its superiority over GPT-2 in generating responses that closely match the standard responses produced by ChatGPT-4. This suggests that LLaMA is more effective in understanding and generating contextually relevant information, leveraging its advanced language modeling capabilities and fine-tuning processes.

Overall, the results demonstrate the potential of LLaMA (LLMs) as a robust model for developing conversational user interfaces that require high accuracy and coherence in response generation, particularly in specialized domains like catalytic converters.

9 Acknowledgments

I would like to thank my supervisor **Prof.Yogesh Kumar Meena** at **IIT Gandhinagar** for their guidance and support throughout this project. Special thanks to the LangChain and Hugging Face communities for their invaluable tools and resources.

References

[1] Build a chatbot. `https://python.langchain.com/v0.2/docs/tutorials/chatbot/`.

[2] Vector stores and retrievers. `https://python.langchain.com/v0.2/docs/tutorials/retrievers/`.

[3] Samuel Kernan Freire, Chaofan Wang, and Evangelos Niforatos. Chatbots in knowledge-intensive contexts: Comparing intent and llm-based systems. *arXiv preprint arXiv:2402.04955*, 2024.

[4] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[5] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[6] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023.