2024 International Seminar on Artificial Intelligence, Computer Technology and Control Engineering (ACTCE) | 979-8-3315-2171-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ACTCE65085.2024.00053

Design and Implementation of LangChain-based Chatbot

Huaxiao Zhang
Century College Beijing University of Posts and Telecommunications
Beijing, China
zhanghuaxiao@ccbup.cn

Zhigang Li*
Century College Beijing University of Posts and Telecommunications
Beijing, China
owent@sina.com

Fang Liu
Century College Beijing University of Posts and Telecommunications
Beijing, China
liufang@ccbup.cn

Yilan He
Century College Beijing University of Posts and Telecommunications
Beijing, China
heyilan@ccbup.cn

Zhichen Cao
Century College Beijing University of Posts and Telecommunications
Beijing,China
caozhicheng@ccbupt.cn

Yunbo Zheng
Century College Beijing University of Posts and Telecommunications
Beijing,China
zhengyubo@ccbupt.cn

Abstract-This paper takes the digital materials such as product data manual and videos as a local knowledge base (LKB), and utilizes the learning ability of large-scale language models (LLMs) to construct a chatbot. It can be used for training new employees or as a technical assistant for on-site maintenance personnel. In addition to providing one-on-one answers to employees through dialogue, it can also search for videos based on keywords and locate relevant information in the video. The LLM driving the chatbot can be either a mainstream online LLM or a locally deployed LLM. The LKB will have priority in the conversation and will only access external models or network resources when the answer cannot be searched from the LKB.

Keywords-LangChain, chatbot, Local Knowledge Base, RAG，video retrieval

I. INTRODUCTION

Currently, rapidly developing artificial intelligence (AI) is supplementing human intelligence. An important branch of AI is conversational AI, which is at the center of the AI revolution. Conversational agent [1] is an artificial intelligence (AI) program, also known as a chatbot, or intelligent robot, intelligent agent, intelligent virtual assistant. They are widely used in business [2] [3] for marketing and customer support in healthcare [4].

II. RESEARCH ON RELATED TECHNOLOGIES

A. Technical Brief

Recently LLMs, especially pre-trained language models (PLMs), have demonstrated remarkable ability to understand and generate cross-lingual texts. These models have demonstrated an impressive level of generalization through self-supervised training on large text corpora [5]. However, LLMs are not yet fully compatible with people's needs in various scenarios for the following reasons. One is that the timeliness of its training cannot cover the latest facts and concepts. For example, Open AI's GPT-3.5 training date is as of September 2021, and GPT-4 training date is as of April 2023. Another is that LLMs do not always work well for tasks in specific specialties. Researchers have proposed different methods to overcome the inherent limitations, such as full fine-tuning [6], Parameter Efficient Fine-Tuning (PEFT), fast engineering, and Retrieval Augmented Generation (RAG) [7].

LangChain, as a development framework for LLMs, aims to take advantage of the inherent capabilities of LLMs to build all kinds of applications [8][9][10]. LangChain supports different models and allows them to interact efficiently with their environment LangChain provides developers with standardized, extensible interfaces and external integration through six core modules, namely, Model I/O, RAG, Chain, Memory, Agent and Callback module.

In the paper, we will use digital materials such as data manuals and videos as external data to build a chatbot to train new employees or serve as a technical assistant for on-site maintenance personnel.

B. Data Connection and Local Knowledge Bases

This section discusses how to connect external data in the LangChain framework. The data manuals and training videos, when used as external data to connect to large models, are referred to as Local Knowledge Bases (LKBs). The LKB not only compensates for the lack of "knowledge" in LLMs, but also makes the developed application more "reliable". This is because such applications can answer questions based on real external data, rather than relying solely on the knowledge learned by the LLMs during training.

External data is often very large, in order to avoid the limitations of the max token, LangChain provides a variety of document converters, which can perform operations such as cutting, combining, filtering, etc. on documents. With them, a long document is cut into a series of small paragraphs, each of which can be used as an input to an independent model.

C. LEDVR Workflow

LangChain's workflow is summarized as LEDVR, as shown in Figure 1.

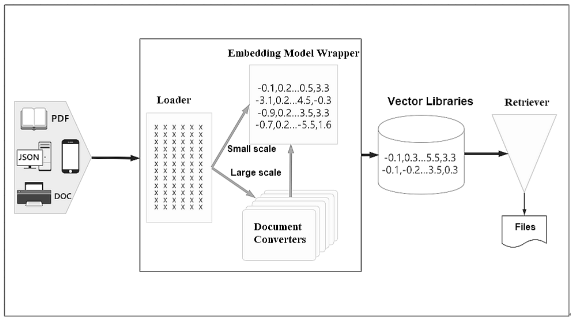

Figure 1. LangChain's workflow.

The loader is responsible for loading external data from various sources into documents, whether it is simple txt or any web page text content. An embedded model wrapper is a class specifically designed to interact with various text embedded models such as OpenAI, Here, Hugging Face, etc. Document converters mainly convert documents through cutting, combining, and filtering. External data is converted into vectors using LangChain's text embedding model and saved in a vector database. By using a vector library, we can also calculate the similarity between two vectors. Retriever is an interface that returns unstructured query documents.

III. DESIGN OF CHATBOT

A. The Requirement Analysis of the Chatbot

According to computer file formats, digital materials include txt, pdf, images, audio, and video. Due to the fact that most documents such as pptx, docx, and txt can be converted to PDF format. Therefore, this article mainly focuses on PDFs and videos. The summary of requirements is shown in Table 1.

Table 1. Summary of requirements of the chatbot.

Number	Requirements	Descriptions	Methods
1	Q&A	Prioritize accessing the local knowledge base. Only use pre trained models or conduct external searches to answer questions when no results are found in the search	External data connection
2	Video Retrieval	Search for video files that match the content from multiple videos, start the player and locate the playback time, such as 2 minutes and 10 seconds	Callback handler
3	Test Questions	Generate relevant test questions based on the digital materials	Memory component

B. Retrieval Augmentation Generation Process

The message sequence of RAG technology is shown in Figure 2, which includes the following steps. Firstly, the user submits a question, and then RAG searches for relevant documents that may answer this question. These documents typically contain proprietary data and are stored in vector databases. RAG constructs some prompt words that combine user input and relevant documents, and guides LLMs to use relevant documents to answer user questions. Finally, LLMs return the answer to the user's question based on the provided context, which is the output of the system.

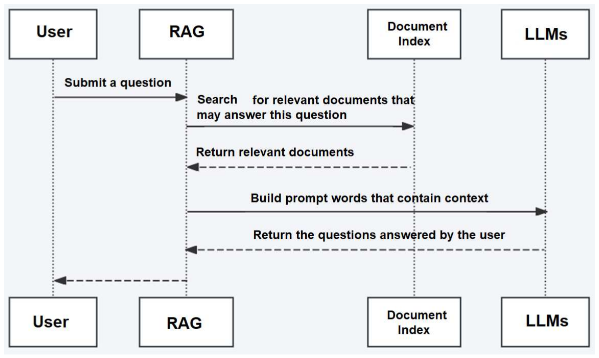
User
RAG
Document Index
LLMs
Submit a question
Search for relevant documents that may answer this question
Return relevant documents
Build prompt words that contain context
Return the questions answered by the user
User
RAG
Document Index
LLMs

Figure 2. RAG application timing diagram.

C. Text Embedding and Cosine Similarity

Text Embedding is a kind of method of mapping high-dimensional discrete numbers (e.g., text) into a low-dimensional continuous vector space. The mapping preserves the semantic relationships between the data, making it easier for machine to understand and process them. Through embedding, we can convert complex textual information into a numerical form that can be processed by computers, thus enabling effective feature representation in a variety of machine learning tasks. The similarity between two vectors (and ultimately two texts) is usually measured using the cosine similarity measure.

Assuming there are two n-dimensional vectors x, y , the simplest way to deal with them is to store them in a flat structured graph, from which their Euclidean distance and cosine similarity expressions are obtained.

Euclidean distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)} \qquad \text{(Formula 1)}$$

Cosine similarity

$$sim(x,y) = \frac{x \cdot y}{x \ \cdot \ y} \qquad \text{(Formula 2)}$$

IV. IMPLEMENTATION OF CHATBOT

A. Handling of Pdf Documents

LangChain provides APIs for Python and JavaScript. We uses Python to implement the requirements in Table 1 and Faiss as a vector library, which is a vector library developed by Facebook (now Meta) for efficiently searching vectors in high-dimensional space. The specific processing of an external pdf document includes the following steps.

1) Load Pdf Documents

The method of pdfReader can be used to load pdf documents into specified variables.

2) Extract the Text of the Pdf Documents

Extract the text of each page of the pdf documents into a string by the method of extract_text, and then divide each page of text into multiple chunks by text_stplitter. The size of each chunk is specified by the parameter chunk_ size, and it is important to ensure that there is a certain overlap between each chunk, which is specified by parameter chunk_ overlap. For example, chunk_ size=1000 and chunk_ overlap=200 indicate that each chunk has a size of 1000 characters, with 200 characters overlapping between them.

3) Text Vectorization and Storage

For the convenience of understanding and processing text, we use OpenAI Embeddings technology to convert each of the above texts into a vector list, and use Faiss to complete vector storage.

4) Similarity Retrieval

The text of the user query is passed to the method of similarity_search, and then based on the similarity between the vectors (Formula 2), the documents that best match the query text are checked in the vector database by the method of dosearch. The results obtained from the search are represented as an array. The array contains multiple document fragments that match the query text, and finally the output is made to the user by the LLM based on the query question and these document fragments.

5) Creation of Q&A Chain

In LangChain, the main function of the chain is to manage the data flow in the application. It links different components together to form a complete data processing flow. Each component is a link in the chain that handles a single task, and they are interdependent to complete their respective tasks in a predetermined order, ultimately completing a complex task. LangChain has several different built-in chains. The stuff chain used in the paper is illustrated as an example. In stuff mode, all relevant document content is submitted to the LLM for processing, and the next step is to build a query. In step 4) above, the content returned from the vector storage is used as a context fragment to answer the query, and then the query is passed to the LLM. After that, the method of chain.run is executed, and the LLM will answer the query and provide relevant answers.

B. Video Retrieval

Assuming some videos have already been deployed on the server. In order to achieve video retrieval, the subtitle files of these videos need to be obtained first. The method of obtaining subtitle files is not the focus of this article and will not be described. Extract the video id, subtitle text, playback duration, the url from the subtitle file corresponding to each video, and combine them into a JSON file.

```
var jsonObj =
{
    "id":"chapter1",
   "Title":"workflow1-6 in chapter1",
    "url":"xxxx"
   "Subtitle-Time":"1\n--:00:00->00:00:02\Hello\n\n2\n\00:00:003-
                  >00:00:08\nToday, I will explain the workflows for
                  chapter 1\n\n.... "
}
```

For the JSON file, the processing is done according to the method similar to the pdf document, including loading, splitting, vectorization, storage and querying.

Unlike handling conversations based on pdf documents, when similar document fragments are retrieved, the LLM does not provide a text reply. Instead, LangChain passes the video's ID and time to the video player and plays the video dynamically. Here, the video player serves as the callback processor for Langchain. LangChain provides the callback handler as a special wrapper mechanism that allows developers to define methods to respond to different lifecycle times. Whenever a specific time is triggered, the corresponding callback handler will be executed.

C. Test Questions and Memory Component

Requirement 3 requires the chatbot to be able to provide employees with test questions. It is best to prepare test questions based on employees' historical conversations and video retrieval, in order to check whether employees have solved these problems. But currently, almost all LLMs have no memory ability. How does the LLM know who this employee is? What questions did he ask before? LangChain provides us with a memory component.

The memory component has two basic operations: read and write. During each run of the chain component, there are two interactions with the memory component. Before running, the memory component retrieves relevant information from its storage space, which may include previous conversation history, user settings, or other data related to the upcoming task. After completing the operation, the generated output information will be written into the memory component. In this way, this information can be read again and used in future runs. For scenarios that require persistent storage of chat records, LangChain's memory component provides a memory component class integrated with the database, such as SQLChainMessage History and MongoDB HatMessage History.

In terms of specific implementation, a chain called ConversationChain needs to be created. Then add a custom memory component of SpaceAntityMemory to it for storage of user information. Then use the methods of load and save to implement read and write operations.

D. User Interface Implementation

In order to enhance the practicality of the chatbot, we use Gradio for interface interaction design, which is an open-source Python library. Through Gradio, we can quickly build a visual and easy-to-use web interface without writing any web front-end code.

V. CONCLUSION

The paper takes LLMs and LangChain framework to build a chatbot with digital materials as LKB. The chatbot can communicate one-on-one with on-site maintenance staff, answer questions based on digital materials, provide testing questions, and also use RAG technology to achieve video retrieval. This article also implemented a basic graphical interactive interface using Gradio. To verify the performance of the conversational robot, we collected 100 digital files for testing. The test shows that the accuracy rate of answering questions is about 85%, which has a certain practicality.

The next step of research is how to deploy a local large-scale model to enhance the information security of digital data.

ACKNOWLEDGMENTS

This work was financially supported by the fund of College's Student Science and Technology Innovation Project. Thank you to every member of my team for their hard work.

REFERENCES

[1] R. Bavaresco, D. Silveira, E. Reis, J. Barbosa, R. Righi, C. Costa, R. Antunes, M. Gomes, C. Gatti, M. Vanzin, S. C. Junior, E. Silva, and C. Moreira, Conversational agents in business: A systematic literature review and future research directions, Comput. Sci. Rev., vol. 36, May 2020, Art. no. 100239, https://doi.org/10.1016/j.cosrev.2020.100239

[2] E. Adamopoulou and L. Moussiades, Chatbots: History, technology, and applications, Mach. Learn. with Appl., vol. 2, Dec. 2020, 976 Art. no. 100006, doi: https://doi.org/10.1016/j.mlwa.2020.100006

[3] Asbjørn Følstad and Marita Skjuve. 2019. Chatbots for customer service: user experience and motivation. In Proceedings of the 1st International Conference on Conversational User Interfaces (CUI '19). Association for Computing Machinery, New York, NY, USA, Article 1, 1–9. https://doi.org/10.1145/3342775.3342784

[4] J. L. Z. Montenegro, C. A. da Costa, and R. da Rosa Righi, Survey of conversational agents in health, Expert Syst. Appl., vol. 129, pp. 56–67, Sep. 2019, doi: https://doi.org/10.1016/j.eswa.2019.03.054

[5] H. Naveed et al., A Comprehensive Overview of Large Language Models. arXiv, Aug. 18, 2023. Accessed: Aug. 22, 2023. [Online]. Available: http://arxiv.org/abs/2307.06435. https://doi.org/10.48550/arXiv.2307.06435

[6] Howard, J.; Ruder, S. Universal Language Model Fine-Tuning for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia, 1 July 2018. [CrossRef]. https://doi.org/10.18653/v1/p18-1031

[7] Radeva, I.; Popchev, I.; Doukovska, L.; Dimitrova, M. Web Application for Retrieval-Augmented Generation: Implementation and Testing. Electronics 2024, 13, 1361. https://doi.org/10.3390/ electronics13071361.

[8] S. Vidivelli, M. Ramachandran, and A. Dharunbalaji "Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion," Comput. Mater. Contin., vol. 80, no. 2, pp. 2423-2442. 2024. https://doi.org/10.32604/cmc.2024.054360

[9] K. Ananthajothi, J. David and A. Kavin, "Cardiovascular Disease Prediction Using Langchain," 2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2024, pp. 1-6, https://doi.org/10.1109/ACCAI61061.2024.10601906

[10] Jeong, Jaemin, Daeyoung Gil, Daeho Kim, and Jaewook Jeong. 2024. "Current Research and Future Directions for Off-Site Construction through LangChain with a Large Language Model" Buildings 14, no. 8: 2374. https://doi.org/10.3390/buildings14082374