Est. 1841 | YORK ST JOHN UNIVERSITY

Pokhrel, Sangita, K C, Bina and Shah, Prashant Bikram (2025) A Practical Application of Retrieval-Augmented Generation for Website-Based Chatbots: Combining Web Scraping, Vectorization, and Semantic Search. Journal of Trends in Computer Science and Smart Technology, 6 (4). pp. 424-442.

Downloaded from: https://ray.yorksj.ac.uk/id/eprint/11412/

The version presented here may differ from the published version or version of record. If you intend to cite from the work you are advised to consult the publisher's version: https://doi.org/10.36548/jtcsst.2024.4.007

Research at York St John (RaY) is an institutional repository. It supports the principles of open access by making the research outputs of the University available in digital form. Copyright of the items stored in RaY reside with the authors and/or other copyright owners. Users may access full text items free of charge, and may download a copy for private study or non-commercial research. For further reuse terms, see licence terms governing individual outputs. Institutional Repositories Policy Statement

RaY

Research at the University of York St John

For more information please contact RaY at ray@yorksj.ac.uk

Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

A Practical Application of Retrieval-Augmented Generation for Website-Based Chatbots: Combining Web Scraping, Vectorization, and Semantic Search

Sangita Pokhrel[1], Bina K C.[2], Prashant Bikram Shah[3]

Department of Computer Science and Data Science, York St John University, London, United Kingdom

Email: [1]s.pokhrel@yorksj.ac.uk, [2]bina.k-c@yorksj.ac.uk, [3]p.shah@yorksj.ac.uk

Abstract

The Retrieval-Augmented Generation (RAG) model significantly enhances the capabilities of large language models (LLMs) by integrating information retrieval with text generation, which is particularly relevant for applications requiring context-aware responses based on dynamic data sources. This research study presents a practical implementation of a RAG model personalized for a Chabot that answers user inquiries from various specific websites. The methodology encompasses several key steps: web scraping using BeautifulSoup to extract relevant content, text processing to segment this content into manageable chunks, and vectorization to create embeddings for efficient semantic search. By employing a semantic search approach, the system retrieves the most relevant document segments based on user queries. The OpenAI API is then utilized to generate contextually appropriate responses from the retrieved information. Key results highlight the system's effectiveness in providing accurate and relevant answers, with evaluation metrics centered on response quality, retrieval efficiency, and user satisfaction. This research contributes a comprehensive integration of scraping, vectorization, and semantic search technologies into a cohesive chatbot application, offering valuable insights into the practical implementation of RAG models.

Keywords: Large Language Models, Retrieval Augmented Generation (RAG), ChatBot, OpenAI API, Natural Language Processing.

DOI: https://doi.org/10.36548/jtcsst.2024.4.007
Received: 03.12.2024, received in revised form: 27.12.2024, accepted: 11.01.2025, published: 20.01.2025
© 2024 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

1. Introduction

Large Language Model (LLM) - powered interactive systems have transformed human-computer interaction by facilitating dynamic and intelligent open conversations. By using tremendous amounts of pre-trained data LLMs like OpenAI's GPT models are excellent at producing human-like language (Brown et al., 2020). The fact that their knowledge base is fixed and set at the time of training, however, this one is their primary drawback. The Retrieval-Augmented Generation (RAG) architecture was developed to get over this limitation by focusing external knowledge retrieval systems with LLMs (Lewis et al.,2020). This hybrid technique improves the relevance and accuracy of models by enabling them to obtain and incorporate current information during interaction.

The current implementation's distinctive characteristic is its ability to actively interact with live website information. The system analyses user-specified webpages dynamically, extracting information, generating embeddings, and using semantic search to produce customized responses in contrast to traditional RAG-based Chabot's that use static or predetermined document collections. The RAG framework improves on LLMs capabilities by combining them along with retrievers that find appropriate additional information to improve responses. The system's flexibility is limited in traditional implementations because sources of knowledge, such as databases, indexed texts, and datasets, remain fixed (Izacard & Grave, 2021). These abilities have recently been improved to dynamic retrieval from additional information. Through the limitations of real-time scraping, embedding creation, and retrieval chains, the system improves user experience by delivering accurate and contextually relevant information.

1.1 Objective

This research aims to develop a chatbot that can effectively answer user queries about a website's content using the RAG approach, integrating web scraping, embeddings, and semantic search. Web scraping ensures current answers, embeddings capture semantic relationships, and semantic search enhances the chatbot's ability to find relevant passages and answer user queries.

1.2 Significance

The proposed Chabot improves user interaction with websites by providing up-to-date information through web scraping, embeddings, and semantic search methodologies. It saves the user time by dynamically scraping the website and recognizing relevant information through semantic search. This search aims to create intelligent interactive systems.

2. Literature Review

The integration of advanced artificial intelligence (AI) techniques, particularly those related to information retrieval and Natural Language Processing (NLP), has greatly advanced the development of chatbots that operate on websites. One such innovation is Retrieval-Augmented Generation (RAG), which blends generative and retrieval-based models to produce responses that are more accurate and appropriate for the context (Lewis et al., 2020). Applications that require dynamic information retrieval from sizable, regularly updated datasets benefit greatly from RAG. By guaranteeing that RAG-based systems have access to current and pertinent data, the integration of web scraping, vectorization, and semantic search technologies is essential to improving system performance (Izacard and Grave, 2020; Karpukhin et al., 2020).

A hybrid framework known as Retrieval-Augmented Generation incorporates outside knowledge sources into inference to improve the response generation process. To function, the RAG model first pulls relevant passages or documents from a knowledge base. Based on the information retrieved and the input query, the model then generates responses (Lewis et al., 2020). With knowledge-intensive tasks, conventional generative models frequently exhibit factual errors and a lack of coherence. To overcome these shortcomings, this method tackles the issues (Guu et al., 2020). In tasks requiring external knowledge, like open-domain question answering and dialogue systems, Lewis et al. (2020) showed that RAG models perform better than conventional sequence-to-sequence models. RAG models are more adaptive to new domains and topics and less dependent on static training data because they can dynamically access updated information by utilizing large-scale retrieval mechanisms (Izacard and Grave, 2021). Additionally, RAG's modular architecture enables flexible integration with different retrieval and generation components, enabling optimization and customization in accordance with application requirements (Lewis et al., 2020).

Web scraping is a vital method for creating the datasets required for RAG models. During the retrieval phase, RAG models access a knowledge base that is populated by automatically extracted data from websites. For applications that require real-time or frequently updated content, Mitchell (2018) emphasizes that web scraping is especially useful for gathering sizable amounts of data that reflect current and up-to-date information. Web scraping provides for the ongoing updating of the knowledge base, which improves the relevance and precision of the chatbot's responses when it comes to website-based chatbots (Sangita et.al., 2023).

A crucial step in enabling machine learning models to process and comprehend text is vectorization, which is the process of transforming textual data into numerical vectors. There have been many vectorization techniques developed, and each has advantages and disadvantages. Reimers and Gurevych (2019) examine various methods such as Word2Vec, Term Frequency-Inverse Document Frequency (TF-IDF), and more modern transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers). For tasks requiring nuanced understanding, like semantic search in RAG models, BERT embeddings have been demonstrated to capture contextual information more effectively.

Rather than depending only on keyword matching, semantic search seeks to enhance information retrieval by comprehending the intent and contextual meaning behind user queries (Huang et al., 2013). Semantic search improves the retrieval part of RAG-based chatbots by finding and bringing up documents that are semantically relevant to the input query; this gives response generation a better foundation (Guu et al., 2020). To quantify the semantic similarity between queries and documents, advanced embedding techniques and similarity metrics must be used (Reimers and Gurevych, 2019). Deep semantic relationships have proven to be particularly well-captured by transformer-based models, leading to more precise and contextually relevant retrieval outcomes (Humeau et al., 2020). Semantic search is integrated into RAG models, which has been demonstrated to improve performance significantly in tasks like conversational AI and open-domain question answering, giving users more accurate and informative responses (Karpukhin et al., 2020). However, there are still difficulties in finding the best way to balance computational efficiency and retrieval accuracy, especially in large-scale and real-time applications (Johnson et al., 2019). The AI content generation technology has been discussed in this research that is about how to use the appropriate language models to make customized features and tools (Pokhrel et.al., 2023) and customized chatbots for

document summarization and question answering using large language models in (Sangita et.al., 2024).

Advanced website-based chatbots that can handle complex and dynamic information needs can be developed with great potential by integrating RAG models with web scraping, vectorization, and semantic search techniques (Izacard and Grave, 2021). These systems have potential, but there are still several obstacles in the way of their effective implementation. As outdated or erroneous information can result in inaccurate responses and erode user trust, it is imperative to ensure the quality and reliability of the scraped data (Mitchell, 2018). Moreover, optimized architectures and resource management techniques are needed to maintain computational efficiency while handling real-time interactions and processing enormous volumes of data (Johnson et al., 2019). Legal and ethical issues are also important, especially when it comes to data privacy and adhering to laws like the General Data Protection Regulation (GDPR) (Voigt and Von dem Bussche, 2017). Concerns about bias, false information, and user privacy must be addressed by ensuring accountability and transparency in AI-driven chatbot responses (Bender et al., 2021).

3. Methodology

The chatbot's system architecture consists of several components, including a web scraper, content parser, text splitter, vector store, retriever chain, and generative model. The web scraper retrieves HTML content from the website, the content parser extracts and cleans the text, and the generative model divides the content into manageable chunks. OpenAI's language models embed these vectors in a database, and the retriever chain searches for relevant content based on the user's query and conversation history. The generative model generates accurate and relevant responses. The Figure 1 depicts the system architecture overview.

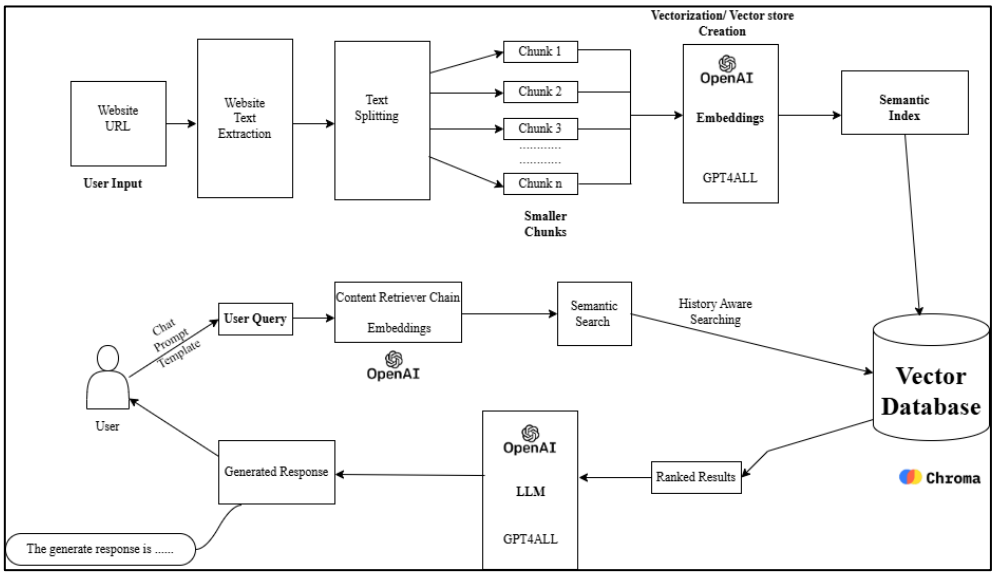

Figure 1. System Architecture Overview

3.1 Data Retrieval

The system utilizes web scraping, BeautifulSoup, and a Python package designed for parsing HTML and XML documents, to extract content from the target website (Richardson, 2022). The *requests* library sends an HTTP request to a website, which is processed by BeautifulSoup, which creates a parse tree to extract individual elements like text, images, and links. BeautifulSoup uses the 'find_all( )' method to locate all instances of an HTML tag extracts the textual content using the 'get_text( )' method to remove extraneous HTML markup (Mitchell, 2018).

3.1.1 Libraries Employed

The system uses Python-based libraries like Streamlit, LangChain, BeautifulSoup4, OpenAI API, and Chroma for tasks like websites data import, embedding creation, and vector database creation. These libraries help with interactive web apps, pipeline administration, HTML extraction, and embedding storage.

3.1.2 Data Extraction/Retrieval

The data extracted from website content obtained through web scraping requires systematic retrieval and parsing, with various methods available once BeautifulSoup has loaded the document. For example, to search for HTML tags by name, attribute, or content, the functions 'find()' and 'find_all()' is used (Richardson 2020). The system can extract

information and navigate the HTML document's intricate structure. BeautifulSoup cleans and organizes extracted data for text splitting and vectorization, ensuring a responsive chatbot that utilizes up-to-date information.

3.2 Text Processing

Splitting text into smaller, more manageable chunks or documents for additional analysis is an essential step in processing extracted content (Mitchell, 2018). This division is required because handling enormous amounts of textual data can be difficult and may result in inefficiencies or problems with content management (Jurafsky and Martin, 2009). The *RecursiveCharacterTextSplitter* is a technique used for text splitting, dividing long texts into smaller chunks within language model's token size limitations, preserving context and ensuring compliance.

3.3 Vectorization

To convert text chunks into dense vector embeddings using a pretrained embedding model, the system uses Chroma for vectorization and OpenAIEmbeddings to create vector representations of text. These vectors are then stored in vector databases (Chroma), which serves the dual purposes of encoding semantic meaning into numeric representations and facilitating effective similarity searches for information retrieval.

3.4 Semantic Search

Reimers and Gurevych (2019) define semantic search as the process of using vector embedding to compare and rank text chunks according to how relevant they are to a user's query. The angle between vectors in the vector space is measured to determine the similarity between them, this comparison is typically carried out using cosine similarity or other distance metrics (Manning, Raghavan, and Schütze, 2008). Higher ranking and retrieval of the most pertinent results are achieved by text chunks that have the highest similarity scores to the query vector. According to Brown et al. (2020), semantic search yields more precise and contextually relevant results than traditional keyword-based searches because it comprehends the context and meaning behind the text. Semantic search is a feature that locates and displays documents that are semantically connected to the user's text query (Jeong., 2023). Semantic search process is shown in the given Figure 2.

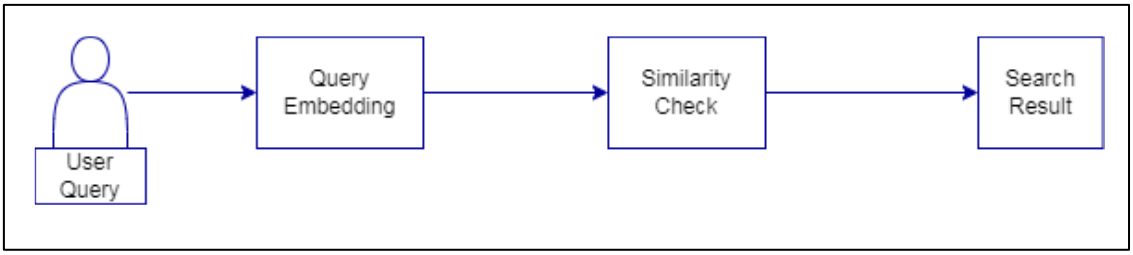

Figure 2. Semantic Search

Semantic search improves user response by analysing user queries and determining their meaning. OpenAI's Chat model processes user input and conversation history creating contextually appropriate queries. ChatPromptTemplate creates search queries based on user intent, retrieving desired information. A retriever chain integrates semantic queries into the search pipeline, utilizing context previous messages and user inquiry for improved retrieval.

3.5 Query Processing

- **Query Embedding**

Text embeddings are strong vector representations of text that effectively compare text similarity by including semantic meaning. The system utilizes OpenAI's text embedding model, including text-embedding-ada-002. The semantic structure of text is represented in high-dimensional space by these embeddings (Brown et al., 2020). For effective embedding and retrieval, the RecursiveCharacterTextSplitter breaks up the text of the webpage into smaller chunks. Multimodal retrieval of relevant information is made possible by the autonomous embedding of each chunk.

- **Similarity Checks**

The Chroma vector storage calculates inverse similarity between user queries and document embeddings, enabling quick and effective similarity-based searches, and transforms user queries into vector storage for query matching. The stored document vector and the query vector are compared using cosine similarity, a lower angle denotes a stronger semantic similarity (Manning et al., 2008). Based on the similarity scores, the top-ranked document chunks are retrieved through the vector_store.as_retriever( ) interface. The language model is then given these pieces to generate responses.

- **Techniques/Algorithm Used**

OpenAI uses conversational GPT model for semantic query processing and prompt engineering for search queries. Pretrained Embedding models transform text into high-dimensional semantic representations. Cosine similarity matches query and document embeddings. RecursiveCharacterTextSplitter splits long texts for retrieval and embedding. Chroma Vector Store saves document embeddings. Figure 3 shows the query processing using embedding.

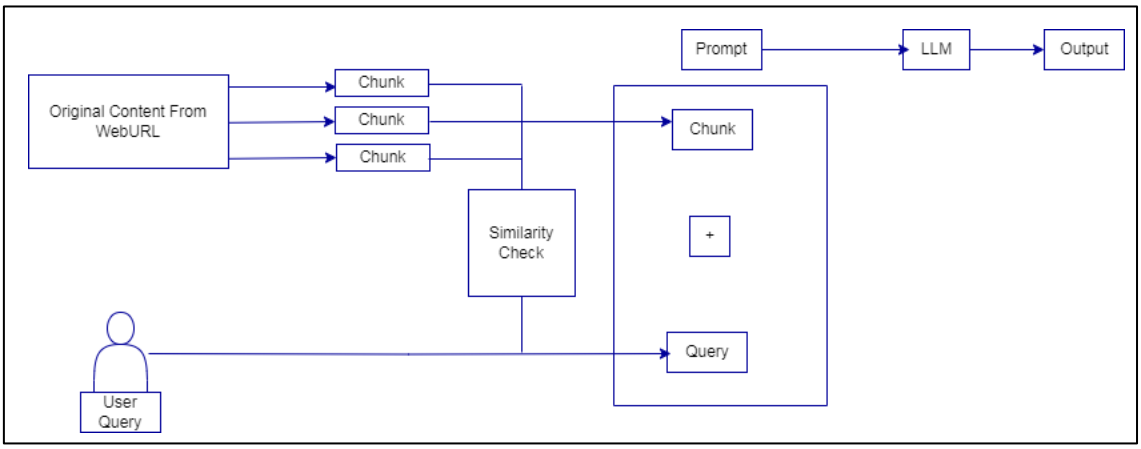

Figure 3. Query Processing using Embedding

3.6 Response Generation

OpenAI's advanced models are used to create an interactive chatbot that dynamically interprets website content, utilizing GPT-4 for conversational responses and text-embedding-ada-002 for efficient embeddings. The retrieved document chunks are integrated into a large language models (LLM), like the OpenAI model, while the response generation process. The LLM uses these chunks to produce a contextually appropriate response to the user's query (Brown et al., 2020). The LLM utilizes semantic information in document chunks and pretrained knowledge to provide precise and user specific responses (Radford et al., 2019). With this method, the model can interpret and rephrase important data in a logical way that is specific to the question at hand, in addition to retrieving important data (Geo et al., 2021).

To ensure that the final response directly and successfully answers the user's query, the output produced by the LLM is refined (Zhang et al., 2019). The process entails identifying the most pertinent segments of the model's response, guaranteeing clarity and consistency, and eliminating any superfluous data that does not aid in addressing the inquiry (Liu et al., 2018). To improve user experience and engagement, the response is then given to the user in an

understandable format, frequently through a conversational interface that mimics natural dialogue (Raffel et al., 2020). This procedure guarantees an easy-to-use and informative product, facilitating a smooth user-system interaction (Brown et al., 2020).

4. Implementation

The application enables conversational querying of web-based content by combining web scraping, text processing, vector database tools, and OpenAI's language model. It involves technical setup, essential procedures (text processing, querying, embedding, and scraping), and addressing development issues. Essential libraries include OpenAI's API, Streamlit, LangChain, BeautifulSoup, and Chroma. Environment variables, including API keys, are managed through dotenv. The Python package manager pip installs necessary libraries, and the application is loaded with these environment variables.

- **Challenges and Solutions**

The implementation of a chatbot application faced several challenges, including handling large content, maintaining conversational context, efficient embedding and retrieval, and managing state in the web interface. LangChain's RecursiveCharacterTextSplitter was used to divide text into manageable chunks, improving the efficiency of the retrieval process. Chroma was used to efficiently organize high dimensional vectors and conduct similarity searches, ensuring timely responses. Streamlit's feature st.session_state was used to maintain chat history and vector store during user interactions. The chatbot's state was managed using Streamlit's feature st.session_state, allowing the chatbot to continue meaningful dialogue by recalling prior exchanges and modifying responses accordingly. A series of tests were conducted to evaluate the effectiveness of the chatbot application's main features, including web scraping, text processing, embedding, semantic search, and the conversational interface.

- **Testing Methods**

The chatbot's performance was evaluated through three steps: website content retrieval, conversational interaction, and response evaluation. The chatbot was given an example URL to retrieve and handle web content, and tested for context preservation, user input comprehension, and data retrieval. The accuracy, coherence, and relevance of the responses were assessed by comparing them with the website's content. The testing results are depicted in Figures 4-9.

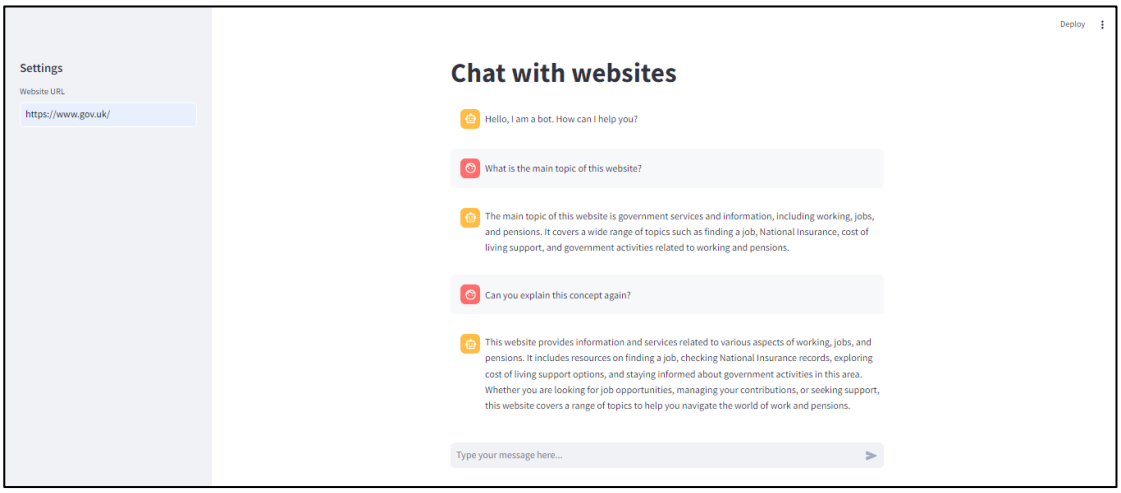

Figure 4. Testing Chatbot General Understanding and Context Awareness

The UK government's website URL was used to test a chatbot's general understanding, context awareness, retrieval accuracy, and complex problem handling. The chatbot was tested on its main purpose, core job features, visa features, and multi-turn conversation, ensuring it could explain visas, benefits for internation students, and compare products or services.

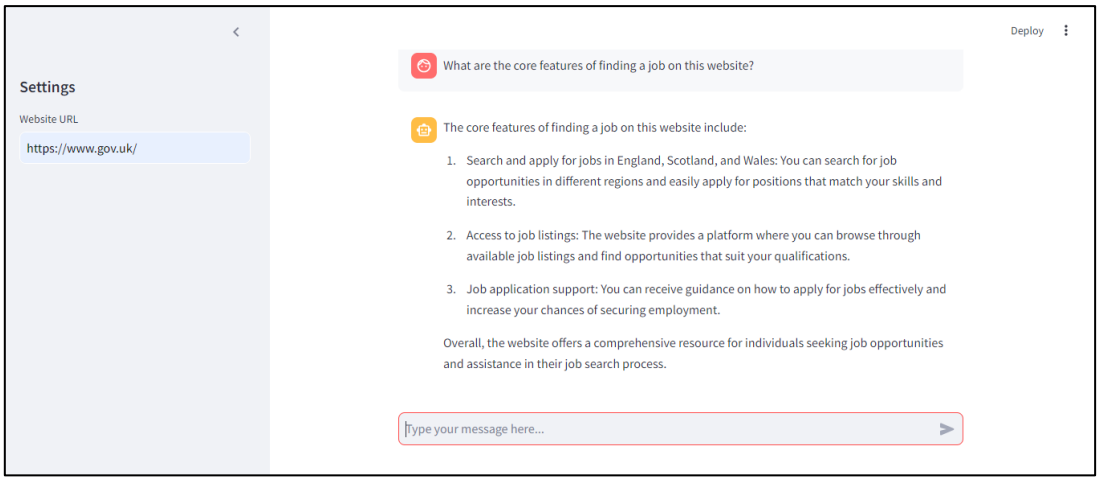

Figure 5. Testing Chatbot's Retrieval Accuracy

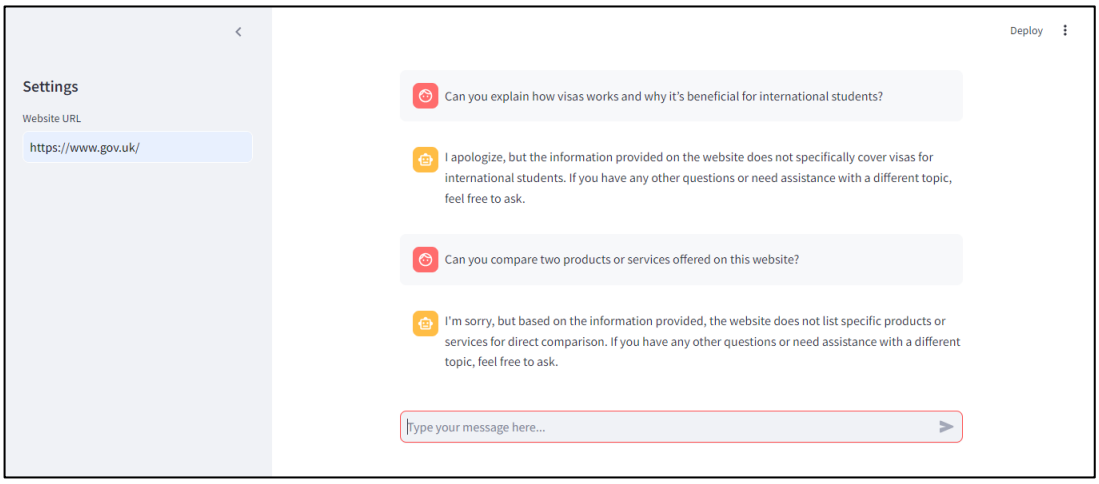

Figure 6. Testing Chatbot's Handling Complex Queries and Multi-Conversation

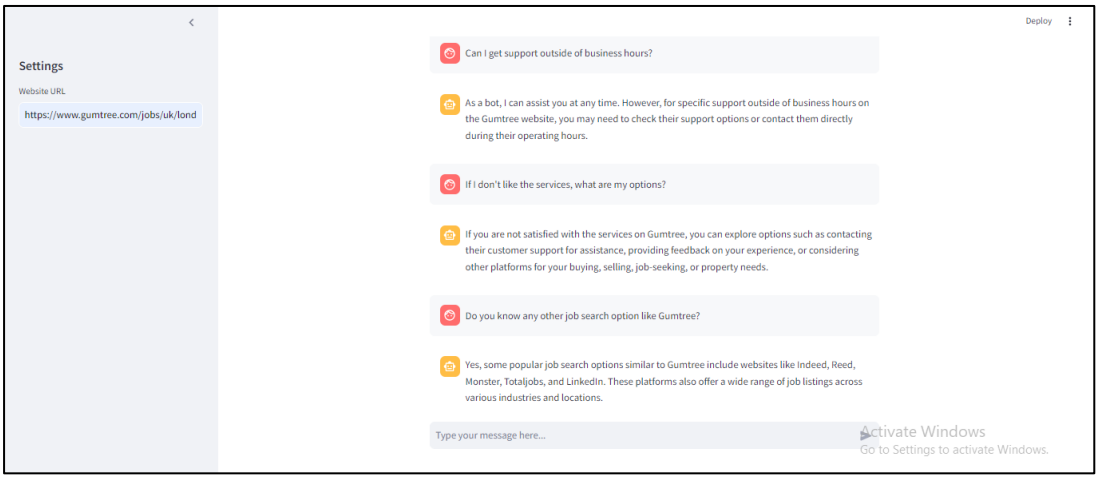

Figure 7. Chatbot's Failure Testing

The chatbot's failure testing was conducted using a website URL and a logistic company URL, while its user experience, response quality, and knowledge boundaries were tested using a random website URL and four questions, generated responses are illustrated in Table 1.

Table 1. Chatbot Responses

Questions	Chatbot Responses
What is this logistic company about? What	This logistics company, Diamond Logistics, specializes in tailored fulfilment and delivery services for individuals and companies. They offer services such as same day delivery, next day delivery, international delivery, UK fulfilment, eCommerce fulfilment, carriers & integrations, and outsourced warehousing and distribution. They also have a logistics

services can they provide?	management platform called Despatch lab that allows you to book, track, and manage your courier requirements and shipments.
What eCommerce fulfilment services they have?	Diamond Logistics offers eCommerce fulfilment services as part of their tailored solutions for businesses. These services include managing inventory, multiple marketplace integrations, and logistics management through their platform, Despatch lab. They ensure transparency, control, and peace of mind for your eCommerce fulfilment needs.
Are your responses based on current information?	Yes, my responses are based on the information provided in the context you shared about Diamond Logistics and its services. If you have any specific questions or need more details, feel free to ask.
Are your responses based on current information?	As an AI assistant, I can provide information and answer questions based on the content available on the website you provided about Diamond Logistics. However, I may not be able to retrieve real-time or updated information that is not included on the website. If you have specific questions beyond the content provided on the website, I may not have access to that information.

We have tested another website which is a website about 10 best supply chain companies of 2021 in this website: https://www.inboundlogistics.com/articles/10-best-supply-chain-companies-of-2021/ and here are the results from the chatbot:

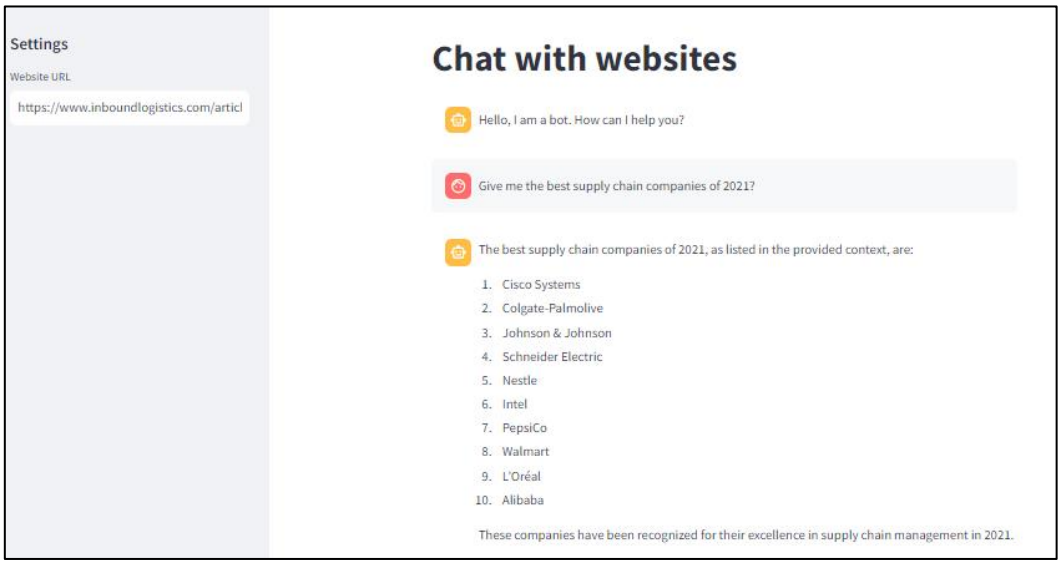

Figure 8. Testing of Chatbot's User Experience, Response quality, Limitations and Knowledge boundaries

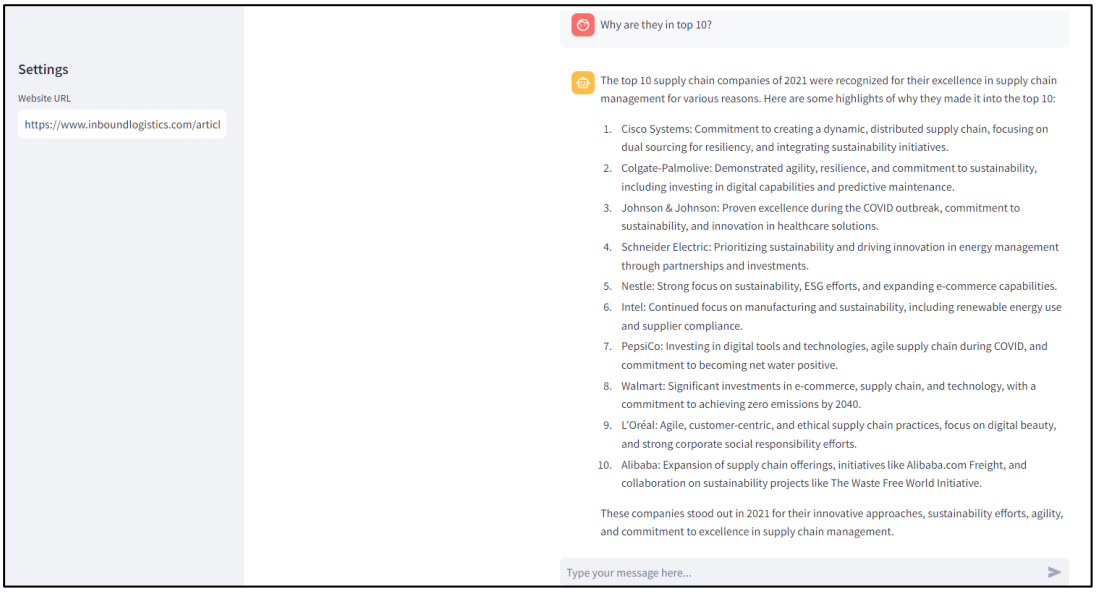

Figure 9(a). Testing Chatbot's Follow – up Question Handling Capacity

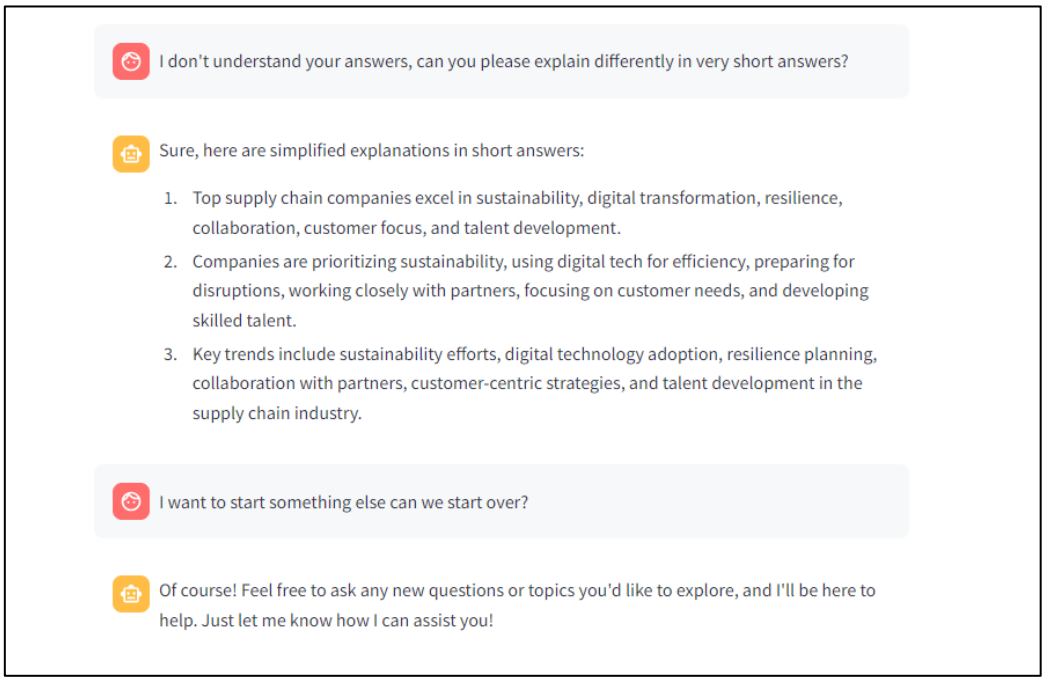

Figure 9(b). Testing Chatbot's Follow – up Questions Handling Capacity

To test the Chatbot's quality to handle Follow-up questions, Figures 9(a) and 9(b) illustrate our chatbot's quality to handle Follow-up questions.

- **Addressing Chatbot Performance Issues**

Problems with embeddings, vector storage, semantic mismatch, and the constraints of the GPT-4 model could make the chatbot's implementation fail. These problems may result in hallucinated responses, confused user enquiries, incomplete or incorrect responses, and improper loading of dynamic website material. For large websites, the GPT-4 model may also produce irrelevant responses and significant latency.

- **Potential Improvements**

To enhance a chatbot's ability to handle complex inquiries and multi-turn dialogues, it can use more complex conversation models, web scraping tools like Selenium, retrieval-augmented generation (RAG) techniques, and a complex query parsing mechanism. These strategies can help the chatbot handle larger contexts, produce more thorough responses, and better understand user intents, especially when dealing with complex instructions or layered information requests.

5. Conclusion and Future Work

The research's main aim is to explore the use of Retrieval-Augmented Generation (RAG) for chatbots on websites. It demonstrates how these methods can retrieve large amounts of data and provide context-relevant information. Web scraping gathers current content, NLP models transform it into embeddings, semantic search connects user queries to relevant content, and a generative model produces logical answers. The RAG model is successful in answering user inquiries, making it practical in dynamic environments like e-commerce and customer support.The RAG chatbot model can be enhanced by adding more languages, incorporating data sources like knowledge bases, social media platforms, or APIs, and creating personalized interaction features. Future work could involve ongoing web scraping, change monitoring, and reinforcement learning to improve the chatbot's intelligence and precision This will ensure users with varying linguistic backgrounds can still benefit from the system.

References

[1] Bender, E.M., Gebru, T., McMillan-Major, A. and Shmitchell, S., 2021, March. On the dangers of stochastic parrots: Can language models be too big?. In Proceedings of the 2021 ACM conference on fairness, accountability, and transparency (610-623). https://doi.org/10.1145/3442188.3445922

[2] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877-1901.

[3] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2019) 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', Proceedings of the 2019 Conferenceof the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, Minnesota, 4171-4186.

[4] Gao, T., Fisch, A., Chen, T., Khashabi, D., and Zettlemoyer, L., 2021. Making Pre-trained Language Models Better Few-Shot Learners. arXiv preprint arXiv:2105.11447.

[5] Guu, K., Lee, K., Tung, Z., Pasupat, P. and Chang, M., 2020, November. Retrieval augmented language model pre-training. In International conference on machine learning (3929-3938). PMLR.

[6] Huang, P.S., He, X., Gao, J., Deng, L., Acero, A. and Heck, L., 2013, October. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management (2333-2338).

[7] Humeau, S., Shuster, K., Lachaux, M.A. and Weston, J., 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. arXiv preprint arXiv:1905.01969.

[8] Izacard, G. and Grave, E., 2020. Leveraging passage retrieval with generative models for open domain question answering. arXiv preprint arXiv:2007.01282.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[10] Jeong, C., 2023. A study on the implementation of generative ai services using an enterprise data-based llm application architecture. arXiv preprint arXiv:2309.01105. Chroma. (2023). Chroma: Vector Database. https://www.trychroma.com/

[11] Johnson, J., Douze, M. and Jégou, H., 2019. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3), 535-547.

[12] Martin, James H. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Pearson/Prentice Hall, 2009.

[13] Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. "Dense passage retrieval for open-domain question answering." arXiv preprint arXiv:2004.04906 (2020).

[14] LangChain. (2023). LangChain Documentation. https://www.langchain.com/docs

[15] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.T., Rocktäschel, T. and Riedel, S., 2020. Retrieval-augmented

generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[16] Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L. and Shazeer, N., 2018. Generating wikipedia by summarizing long sequences. arXiv preprint arXiv:1801.10198.

[17] Manning, C.D., Raghavan, P. and Schütze, H., 2008. Introduction to Information Retrieval. Cambridge: Cambridge University Press.

[18] Microsoft (2023) 'Retrieval Augmented Generation using Azure Machine Learning prompt flow'. Available at: https://learn.microsoft.com/en-us/azure/machine-learning/concept-retrieval-augmented-generation?view=azureml-api-2 (Accessed: 12 September 2024).

[19] Mitchell, Ryan. Web scraping with Python: Collecting more data from the modern web. O'Reilly Media, 2018.OpenAI. (2023). OpenAI API. https://openai.com/api/

[20] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I., 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[21] Pokhrel, Sangita, and Shiv Raj Banjade. "AI Content Generation Technology based on Open AI Language Model." Journal of Artificial Intelligence and Capsule Networks 5, no. 4 (2023): 534-548

[22] Pokhrel, Sangita, Swathi Ganesan, Tasnim Akther, and Lakmali Karunarathne. "Building Customized Chatbots for Document Summarization and Question Answering using Large Language Models using a Framework with OpenAI, Lang chain, and Streamlit." Journal of Information Technology and Digital World 6, no. 1 (2024): 70-86

[23] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

[24] Reimers, N. and Gurevych, I., 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. arXiv preprint arXiv:1908.10084.

[25] Richardson, L. (2022). Beautiful Soup Documentation. [Online]. Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ .

[26] Streamlit. (2023). Streamlit: The Fastest Way to Build Data Apps. https://streamlit.io/

[27] Vaswani, A., 2017. Attention is all you need. Advances in Neural Information Processing Systems. arXiv:1706.03762v7 [cs.CL]

[28] Pokhrel, Sangita, Nalinda Somasiri, Rebecca Jeyavadhanam, and Swathi Ganesan. "Web Data Scraping Technology Using Term Frequency Inverse Document Frequency to Enhance the Big Data Quality on Sentiment Analysis." International Journal of Electrical and Computer Engineering 17, no. 11 (2023): 300-307.

[29] Voigt, P. and Von dem Bussche, A. (2017) The EU General Data Protection Regulation (GDPR): A Practical Guide. 1st Edition, Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-57959-7

[30] Zhang, Y., Sun, S., Galley, M., Chen, Y.C., Brockett, C., Gao, X., Gao, J., Liu, J. and Dolan, B., 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. arXiv preprint arXiv:1911.00536.