Leveraging Structured Data Input for Effective Chatbot Integration in Enterprises

Caio Siqueira[1], **Orlando Guilarte**[1], **Giuliano Ferreira**[1], **Omar Leiva**[1]

[1]Diretoria de Sistemas de Informação - PUC-Rio
Rua Marquês de São Vicente 225 – 22451-900 - Gávea - Rio de Janeiro, Brasil

{csiqueira,ofonsek,giuliano.biblioteca}@puc-rio.br

omarleivac@aluno.puc-rio.br

Abstract. *This paper introduces an approach for integrating structured data into chatbot applications. Utilizing our Mindmap tool, which hierarchically organizes data and maps nodes to actions, we developed an augmented JSON schema to improve chatbot contextual understanding and response accuracy. By applying the Langchain suite and Retrieval-Augmented Generation techniques, our method enhances data retrieval and processing from a vector store, significantly improving interaction relevance.*
Keywords: *Chatbot Integration; Structured Data; RAG; Langchain*

1. Introduction

In recent years, the evolution of consumer-facing software has led to heightened expectations among corporate users for more intuitive and natural interactions with computer systems. Traditionally, user interfaces in corporate environments have relied on window-based interaction paradigms. However, the advent of sophisticated natural language processing (NLP) technologies has begun to shift this paradigm, making natural language interfaces increasingly desirable for enterprise applications [Weiying et al. 2019]. A significant milestone in this transition was the public release of ChatGPT by OpenAI [OpenAI 2023]. The underlying technology behind ChatGPT is based on a Large Language Model (LLM), a type of machine learning model trained on extensive datasets containing diverse textual content. The scale of these training datasets allows LLMs to effectively respond to a wide range of user inputs. However, deploying LLMs in corporate environments presents unique challenges. A key limitation is that these models are typically not trained on proprietary data from private institutions, which must remain confidential to protect organizational integrity and data privacy. To address this issue, researchers have explored the use of knowledge graphs to further enhance LLMs [Wen et al. 2023], and techniques such as Retrieval-Augmented Generation (RAG) have been developed to provide LLMs with context derived from private datasets, thereby enhancing their relevance and accuracy in enterprise settings [Lewis et al. 2020]. Despite the effectiveness of these techniques, they often fall short of meeting the demands of corporate software development, where frequent updates and rapid responses to organizational changes are common. These approaches also face challenges such as difficulty in incorporating new knowledge and explaining their reasoning processes [Wen et al. 2023].

This study proposes a mechanism for structuring data more effectively to feed LLM-based chatbots with the contextual information necessary to provide accurate and

contextually relevant responses within corporate environments. Given the stringent requirements of corporate settings, where control over generated outputs is paramount, the approach outlined in this work prioritizes supervised interaction models over autonomous agent-based systems. Unlike general public applications, where disclaimers can mitigate the risks of inaccurate or flawed outputs, corporate environments require a higher level of oversight to prevent potential adverse outcomes.

2. Conception

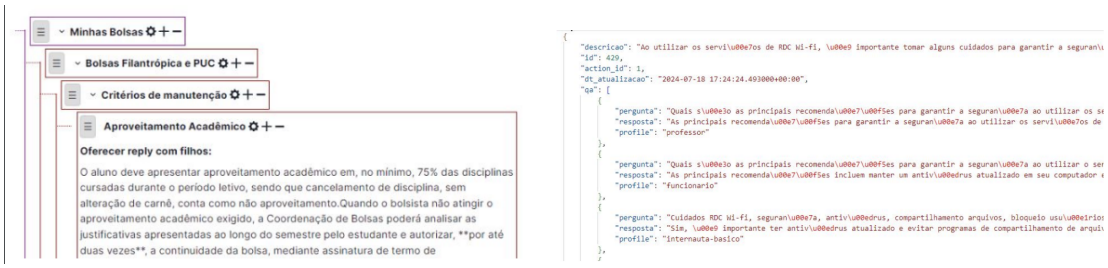
With the growing demand from our key clients, particularly those forming the consumer base of our business model, for a more integrated and sophisticated LLM chatbot interface, we developed an approach that leverages existing structured chatbot systems and the data already mapped within these systems.

The structured chatbot system organized data into a hierarchical, tree-like structure, where each parent node represented a topic, and the child nodes indicated possible responses or subtopics associated with that topic. This setup ensured a well-defined and navigable dialogue structure. With the goal of creating an LLM-based chatbot integrated with RAG — which employs a vector store for semantic retrieval — we developed a tool capable of exporting this structured data for integration into the new system.

However, we identified significant limitations in our current software, which only supported data generation in RTF (Rich Text Format), a format unsuitable for our needs. To overcome this challenge, we developed a new tool to manage the registration of nodes and their associated child nodes. This led to the creation of a web-based interface (Mindmap), built using advanced JavaScript frameworks (React with Next.js [Vercel, Inc. 2024]).

The introduction of this tool significantly expanded the scope of the project, allowing us to move beyond simple topic and child text nodes, enabling the creation of more complex data structures tailored to specific objectives. One of the critical features developed was the mapping of nodes to actions, which dictate the operations to be executed by the chatbot. The text registered within each node is subsequently parsed as parameters for the corresponding action.

With the development of the Mindmap tool, we now have a robust platform that allows for the systematic registration of necessary data and facilitates its export into formats more suitable for subsequent processing.



(a) Screenshot of the Mindmap tool

(b) Snippet of the augmented json

Figure 1. Generated augmented json

3. Proposed pipeline

The Mindmap tool exports hierarchical data as a structured JSON file, which is processed into an augmented JSON using Python scripts within the Langchain suite [LangChain Documentation], currently associated with the GPT-4 model from OpenAI. This augmented file incorporates questions and answers, derived from the hierarchy of each node, its actions, and related metadata.

These questions and answers are generated based on the node's actions and predefined profiles. The resulting JSON is then used to populate a vector store database, serving as the foundation for the chatbot's RAG functionalities. The design is agnostic concerning the chatbot engine's method of consuming this data to construct its vector store or run the inference itself. For example, a collaborating research group utilizes the vector store without incorporating questions and answers, using them later to validate the generated dataset. In contrast, internally, we employ the questions and answers to create documents directly in the vector store. A comparative study on the efficacy of these strategies for various use cases could be the focus of future research.

Table 1. Sample of node actions

Action name	Action description
Fetch data on a website	The node contains a URL and a CSS selector to fetch data. Additionally, it contains a JUDGE text to check if the data is similar to what is expected.
Use a vector store for document	Given a document link, build a vector store specific to that document. Once an initial query matches that the answer should come from the document, a second LLM query is used on that specific vector store.
Fetch an API	Translate the user input into an API call. Translate the return into readable output to serve for the user. The node has instructions on how this should be done.
Serve node text	Use the node text as context to provide an answer to the user input.

4. Data structure

The output augmented JSON schema encapsulates several key elements, including the list of actions within the exported object, the profiles used for generating questions and answers, and the hierarchical structure of nodes and their children. Each node is assigned a unique identifier and an update timestamp, generated by our Mindmap application, to facilitate efficient updates in downstream consumer applications.

To enhance the functionality of each node, we introduced a supplementary structure termed "helper." Each action within a node is associated with a helper, which consists of parameters parsed during the generation of the augmented JSON. Internally, our team loads these helpers into the vector database, enabling their retrieval at runtime via Langchain and Python.

Looking forward, a significant improvement on our roadmap involves integrating the Mindmap frontend tool with the Python backend responsible for generating the

augmented JSON structure. We also plan to migrate the generated content, including questions and answers, into our relational database alongside the nodes. This integration will streamline the workflow, allowing for direct export of the augmented JSON from the Mindmap application, thereby enhancing the user experience and operational efficiency.

Table 2. Sample profiles used when generating questions and answers

Question Profile	Profile description	Used in
Computer science student	You are a computer science student who focuses your input using direct messages	Questions
Language student	You are a language student with rich vocabulary	Questions
Internet user	You are an unknown internet user with poor grammar which basically uses keywords when interacting with systems	Questions
Institutional chatbot	You are an organization chatbot, which needs to answer in a formal way never betraying the ideals of the organization	Answers

5. Conclusion

The work we present proposes an innovative and cohesive approach to integrating the entire workflow in creating a chatbot application that is closely coupled with its underlying data. By leveraging structured data input and the advanced functionalities of our Mindmap tool, we have established a solid foundation that not only supports the continuous generation of enhanced JSON structures but also facilitates real-time data retrieval and processing through advanced technologies such as Langchain and Python.

The management of the Mindmap software is designed to be in the hands of the process owners, ensuring that those who understand the intricacies of each process are directly involved in its configuration and oversight. For example, one of our chatbot instances is currently managed by our Process Management Office (EP) [1].

Our collaboration with the partner research group, the Applied Computational Intelligence Laboratory (ICA) [2], has proven invaluable, providing critical insights and valuable feedback for the continuous improvement of our tool.

Looking ahead, we are confident that the continued development and refinement of this tool will further enhance our ability to integrate complex data structures with chatbot applications, ultimately contributing to more intelligent and responsive systems across the organization. This work not only demonstrates the potential of structured data integration but also lays the groundwork for future innovations in the field of enterprise chatbot solutions.

[1]Escritório de Processos - `https://ep.dsi.puc-rio.br`

[2]Laboratório de Inteligência Computacional Aplicada - `https://ica.ele.puc-rio.br`

References

LangChain Documentation. Langchain: Building applications with llms through composability. n.d.

Lewis, P., Oguz, B., Rinott, R., Riedel, S., and Stenetorp, P. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada.

OpenAI (2023). Gpt-4 technical report. Technical report, OpenAI.

Vercel, Inc. (2024). *React Foundations: About React and Next.js*. Next.js Documentation.

Weiying, K., Pham, D. N., Eftekharypour, Y., and Pheng, A. J. (2019). Benchmarking nlp toolkits for enterprise application. In *PRICAI 2019: Trends in Artificial Intelligence: 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26-30, 2019, Proceedings, Part III 16*, pages 289–294. Springer.

Wen, Y., Wang, Z., and Sun, J. (2023). Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*.